Hierarchical feature extraction from spatiotemporal data for cyber-physical system analytics

by

Adedotun John Akintayo

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering (Applied Machine Learning)

Program of Study Committee:
Soumik Sarkar, Co-major Professor
Baskar Ganapathysubramanian, Co-major Professor
Theodore John Heindel
Asheesh Kumar Singh
Namrata Vaswani

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2017

DEDICATION

This research is dedicated to: the glory of God the father, son and spirit for granting the ability to complete the studies; the memory of my late grandparents who, though were not educated, found it worthy to bequeath the best legacy to my parents Dr. & Mrs. Akintayo, who have also given me and my siblings Mr. Akinyemi Akintayo, Mrs. Abisola Adedoyin and Mr. Akinjide Akintayo the best parental support; my jewel of inestimable worth Olubukola Afolabi for enduring years without my presence in Nigeria.

Finally, I dedicate the presentation aspect to the memory of a late 5-month-old baby Miracle of Martha and Kevin who had survived several surgeries, but gave up the ghost in the week of my final defense. Miracle's strength in the struggles was the final inspiration for my successful defense.

TABLE OF CONTENTS

Page

LIST OF TABLES x

LIST OF FIGURES xii

ACKNOWLEDGMENTS xxiii

ABSTRACT xxiv

CHAPTER 1. OVERVIEW 1

1.1 Introduction 2

1.2 Research Goal 3

1.2.1 Spatial data applications 4

1.2.2 Temporal data applications 6

1.2.3 Spatiotemporal data applications 7

1.3 Contributions 9

1.4 Organization of Dissertation 11

CHAPTER 2. REVIEW OF LITERATURE 13

2.1 Introduction 13

2.1.1 Learning model - REO 14

2.1.2 Learning model - TPE 15

2.2 Learning and Modeling Paradigms 15

2.2.1 Supervised, semi-supervised and unsupervised learning 16

2.2.2 Generative and discriminative models 17

2.3 Machine Learning Procedure . . . 18
2.3.1 Data pre-processing . . . 18
2.3.2 Feature extraction . . . 19
2.3.3 Model training & inference . . . 20
2.4 Hierarchical Feature Extraction (HFE) . . . 21
2.4.1 Spatial data . . . 22
2.4.2 Temporal data . . . 23
2.4.3 Spatiotemporal data . . . 24
2.5 Development of Neural Networks . . . 26
2.5.1 Loss functions . . . 32
2.5.2 Optimization schemes . . . 34
2.5.3 Backpropagation algorithm . . . 42
2.6 Learning Architectures . . . 43
2.6.1 Stacked sparse denoising autoencoder (SSDA) . . . 43
2.6.2 Convolutional networks . . . 44
2.6.3 Autoencoders . . . 47
2.6.4 Convolutional autoencoders . . . 49
2.7 Temporal Data Analytics . . . 51
2.7.1 Discretization and symbolization . . . 51
2.7.2 Symbolic dynamic filtering technique . . . 52
2.7.3 xD-Markov machines extension of SDF . . . 54
2.8 Summary . . . 56
CHAPTER 3. FEATURE EXTRACTION FROM SPATIAL DATA – LOW LIGHT NETWORK (LLNet) . . . 57
3.1 Introduction . . . 57
3.1.1 Motivation . . . 58

3.2 The Low-light Net (LLNet) . . . 59
3.2.1 Learning features from low-light images with LLNet . . . 60
3.2.2 Network parameters . . . 61
3.2.3 Training the model . . . 62
3.2.4 Image reconstruction . . . 64
3.3 Evaluation metrics and compared methods . . . 64
3.3.1 Performance metric . . . 64
3.3.2 Compared methods . . . 66
3.4 Results and Discussion . . . 68
3.4.1 Algorithm adaptivity . . . 68
3.4.2 Enhancing artificially darkened images . . . 70
3.4.3 Enhancing darkened images in the presence of synthetic noise . . . 70
3.4.4 Application on natural low-light images . . . 72
3.4.5 Denoising capability, image sharpness, and patch size . . . 72
3.4.6 Prior knowledge on input . . . 73
3.4.7 Features of low-light images . . . 74
3.4.8 Hyper-parameters, network architecture, and performance . . . 75
3.5 Summary . . . 76
CHAPTER 4. FEATURE EXTRACTION FROM SPATIAL DATA – CONVOLUTIONAL SELECTIVE AUTOENCODER (CSAE) . . . 81
4.1 Introduction . . . 81
4.2 Selectivity in Deep Networks . . . 81
4.2.1 Mathematical formulation of selectivity . . . 82
4.2.2 Convolutional selective autoencoder (CSAE) . . . 84

4.3 An End-to-end Convolutional Selective Autoencoder Approach to *Soybean Cyst Nematode* Eggs Detection . . . 85
4.3.1 Motivation . . . 86
4.3.2 Algorithm . . . 87
4.3.3 Dataset and implementation . . . 89
4.3.4 Results and discussions . . . 91
4.4 Prognostics of Combustion Instabilities from Hi-speed Flame Video using A Deep Convolutional Selective Autoencoder . . . 98
4.4.1 Motivation . . . 99
4.4.2 Algorithm . . . 104
4.4.3 Dataset and implementation . . . 106
4.4.4 Threshold determination . . . 109
4.4.5 Results and Discussions . . . 109
4.4.6 Early detection . . . 115
4.5 Conclusion . . . 119
CHAPTER 5. FEATURE EXTRACTION FROM TEMPORAL DATA – HIERARCHICAL FEATURE EXTRACTION . . . 120
5.1 Introduction . . . 120
5.2 Background and Motivation . . . 121
5.2.1 The CRP distribution and stickiness adjustment . . . 124
5.3 Proposed Hierarchical SDF Methodology . . . 125
5.3.1 Data Likelihood estimation . . . 126
5.3.2 Assignment of a slow time-scale behavior to an existing or new class . . . 129
5.3.3 Computational complexity of HSDF . . . 133
5.3.4 Off-line PFSA revision . . . 133
5.4 Improvement of Data Likelihood . . . 134
5.4.1 Preliminaries . . . 135

5.5 Validation Results and Discussion . . . 136
5.5.1 Simulated nonlinear dynamical systems . . . 137
5.5.2 Results and performance comparison . . . 138
5.5.3 Real use case of energy disaggregation . . . 144
5.5.4 Ground truth formulation, results and performance comparison . . . 145
5.6 Summary . . . 150
CHAPTER 6. FEATURE EXTRACTION FROM SPATIOTEMPORAL DATA – SPATIOTEMPORAL PATTERN NETWORK (STPN) . . . 153
6.1 Introduction . . . 153
6.2 Background and Motivation . . . 154
6.3 Spatiotemporal Pattern Network . . . 155
6.3.1 Symbolic modeling of dynamical systems and interactions . . . 155
6.4 Supply Side: Wind Turbines . . . 159
6.4.1 Geographical information . . . 159
6.4.2 Results and discussion . . . 163
6.5 Demand Side: Non-Intrusive Load Monitoring . . . 170
6.5.1 Problem description . . . 170
6.5.2 Compared techniques . . . 171
6.5.3 Results and discussion . . . 172
6.6 Summary . . . 183
CHAPTER 7. FEATURE EXTRACTION FROM SPATIOTEMPORAL DATA – 3D-CONVOLUTIONAL SELECTIVE AUTOENCODER (3D-CSAE) . . . 184
7.1 Background and Motivation . . . 184
7.1.1 3D temporal stacking of 2D frame sequence . . . 187
7.2 Approach . . . 189
7.2.1 3D convolutional autoencoder (3D-CSAE) . . . 189
7.2.2 Network parameters and structure . . . 192

7.2.3 Training . . . 193
7.2.4 Experiments . . . 194
7.2.5 Evaluation metric and threshold . . . 195
7.3 Results and Discussion . . . 195
7.3.1 Visualization of layer-wise class activations . . . 195
7.3.2 Testing protocols . . . 196
7.4 Summary . . . 203
CHAPTER 8. SUMMARY, CONCLUSION AND FUTURE RESEARCH DIRECTION . . . 204
8.1 Summary and Conclusion . . . 204
8.1.1 Feature extraction from spatial datasets . . . 205
8.1.2 Feature extraction from temporal datasets . . . 205
8.1.3 Feature extraction from spatiotemporal datasets . . . 206
8.2 Future Research Areas . . . 206
8.2.1 Generative adversarial networks (GAN) . . . 206
8.2.2 Recurrent neural networks (RNNs) and long short term memory networks (LSTMs) . . . 208
APPENDIX A. ARCHITECTURAL PLATFORM AND HARDWARE . . . 209
A.1 Introduction . . . 209
A.2 GPU Computing Libraries . . . 210
A.2.1 Open Multi-Processing (OpenMP) and Open Computing Library (OpenCL) . . . 211
A.2.2 Compute Universal Device Architecture (CUDA) . . . 212
A.3 GPU Hardware . . . 215
A.4 Summary . . . 217
APPENDIX B. ALGORITHM IMPLEMENTATION SOFTWARE . . . 218
B.1 Trending Deep Learning Platforms . . . 218
B.1.1 Caffe . . . 218
B.1.2 Theano . . . 219

B.1.3 TensorFlow 220
B.1.4 Torch 221
B.1.5 Keras 222
B.2 Data and Model Parallelism in GPUs 223
B.3 Summary 224
BIBLIOGRAPHY 225

LIST OF TABLES

Page

Table 3.1 PSNR and SSIM of outputs using different enhancement methods. 'Bird' means the non-dark and noiseless (i.e. original) image of Bird. 'Bird-D' indicates a darkened version of the same image. 'Bird-D+GN18' denotes a darkened Bird image with added Gaussian noise of $\sigma = 18$, whereas 'Bird-D+GN25' denotes darkened Bird image with added Gaussian noise of $\sigma = 25$. Bolded numbers corresponds to the method with the highest PSNR or SSIM. Asterisk (*) denotes our framework. 69

Table 3.2 Average PSNR evaluated on the set of natural dark images enhanced with trained model of different hyper-parameters and network architecture. The baseline model is marked by an asterisk (*), whereas the PSNR for the best model is presented in a bolded typeface. 'Btk' is an abbreviation of 'bottleneck' that describes a model that has a decreasing number of hidden units toward the center. 75

Table 4.1 Training set breakdown: cropped – S, translated – T, rotated – R and labeled – L. 91

Table 4.2 Detection time required for an image frame with different patch sizes. 95

Table 4.3 Performance metrics for SCN egg detection problem. 96

Table 4.4 Breakdown of the experimental dataset used for training the CSAE and SCRF algorithms showing how different combination of factors result in either stable or unstable states. 108

Table 4.5 CSAE optimum encode layer size metric and transition start frame # for protocol 500_{40to30}. 113

Table 4.6 CSAE and SCRF comparison 114

Table 4.7 Performance metrics and transition start frame # for transition protocols. . 116

Table 5.1 Performance comparison of algorithm versions under different noise levels. . 141

Table 5.2 Performance comparison of HSDF and HDP-HMM approaches on simulated datasets 144

Table 5.3 Performance comparison of HSDF and HDP-HMM approaches on low frequency REDD 151

Table 6.1 Computational information for different methods in April. 179

Table 6.2 Performance (MSE) of STPN with comparison to FHMM and CO. 180

Table A.1 Available facilities in Self-aware Complex Systems laboratory in September 2017 216

LIST OF FIGURES

Page

Figure 1.1 Example of a Degraded Visual Environment (DVE) - dusty condition ("courtesy: www.ensco.com"). 5

Figure 1.2 High dimensional speech signals of a set of business people in a meeting: Courtesy of "Cartoon resources on Google". 7

Figure 1.3 Automated aerial imagery for pattern recognition and efficient crop management. 8

Figure 2.1 Machine learning applied to engineering problems with a focus on hierarchical feature extraction. 14

Figure 2.2 A simple perceptron training rule schematics. 28

Figure 2.3 Artificial neural network activation examples: (a) rectified linear unit ($\sigma_{relu} = max(0, z)$) (b) sigmoidal activation ($\sigma_{sigmoid} = \frac{1}{1+e^{-z}}$) and their respective Gaussians depicting their relative activation strengths around the values. . . 30

Figure 2.4 Schematics showing comparison between, (a). wider network and (b). deeper network. 31

Figure 2.5 Data sampling techniques available for the stochastic gradient descent algorithm. 35

Figure 2.6 Local and global minima in dots, and their respective starting points are starred on an example function. 37

Figure 2.7 Schematics of different fits of an arbitrary function: $cos(1.5\pi x)$ plus random noise with, (a). underfitting – linear fit (b). approximate fit – 2^{nd} degree polynomial fit, and (c). over-fitting – 15^{th} degree polynomial fit. 40

Figure 2.8 Schematics of the convolutional layers. 45

Figure 2.9 Schematics of the pooling layer. 46
Figure 2.10 Encode layer for cnn classification 47
Figure 2.11 Pictorial description of the fully connected layers that make up an autoencoder network 48
Figure 2.12 Schematics of the upsampling layer that distinguishes a convolutional autoencoder from a cnn. 50
Figure 2.13 Illustration of generation of a PFSA using (a) maximal bijectively discretization and (b) maximum entropy partitioning for system A. 53

Figure 3.1 Architecture of the proposed framework: (a) An autoencoder module is comprised of multiple layers of hidden units, where the encoder is trained by unsupervised learning, the decoder weights are transposed from the encoder and subsequently fine-tuned by error back-propagation; (b) LLNet with a simultaneous contrast-enhancement and denoising module; (c) S-LLNet with sequential contrast-enhancement and denoising modules. The purpose of denoising is to remove noise artifacts often accompanying contrast enhancement. 60
Figure 3.2 Training the LLNet: Training images are synthetically darkened and added with noise. These images are fed through LLNet where the reconstructed images are compared with the uncorrupted images to compute the error, which is then back-propagated to finetune and optimize the model weights and biases. 65
Figure 3.3 Original standard test images used to compute PSNR. 70
Figure 3.4 Comparison of methods of enhancing 'Town' when applied to (A) original already-bright, (B) darkened, (C) darkened and noisy ($\sigma = 18$), and (D) darkened and noisy ($\sigma = 25$) images. Darkening is done with $\gamma = 3$. The numbers with units dB are PSNR, the numbers without are SSIM. 71

Figure 3.5 Comparison of methods of enhancing naturally dark images of (A) computer, (B) chalkboard, (C) cabinet, and (D) writings. Selected regions are enlarged to demonstrate the denoising and local contrast enhancement capabilities of LLNet. HE (including HE+BM3D) results in overamplification of the light from the computer display whereas LLNet was able to avoid this issue. In the natural case, the best performers in terms of PSNR and SSIM coincide. 77

Figure 3.6 Relative patch size vs PSNR and SSIM. The picture with highest PSNR has the highest denoising capability but least sharp. Picture with lowest r has the least denoising capability but has the highest image sharpness. Picture with the highest SSIM is more well-balanced. 78

Figure 3.7 Evaluation on US Air Force (USAF) resolution test chart. There exist optimal relative patch sizes that result in the highest PSNR or SSIM after image enhancement (using LLNet). Note that the result enhanced with histogram equalization is shown to highlight the loss in detail of the natural dark image (where the main light is turned off) compared to the natural bright image. . 79

Figure 3.8 Feature detectors can be visualized by plotting the weights connecting the input to the hidden units in the first layer. These weights are selected randomly. 80

Figure 3.9 Random selection of weights from the first layer (feature detectors) and weights from the output layer (feature generators) for the integrated LLNet model trained with a patch size of 21×21. Patterns in the output weights are similar to patterns in the first hidden layer weights since tied weights are used. 80

Figure 3.10 Random selection of first layer weights from an integrated LLNet model trained in batches of 1000 and 50, respectively. The superior model (i.e. batch size 50) learns features that appear more distinctive. 80

Figure 4.1 Plate a) Shows an example of a frame with eggs in purple boxes and unwanted particles on a *MATLAB*-based GUI for tool-labeling of plant pathologists-identified eggs and Plate b) shows the ground truth labels derived. 87

Figure 4.2 Convolutional autoencoder architecture. Letters are used to describe the layer types, c- convolution, m-maxpooling, r – reshape, u – unpooling and d – deconvolution layers respectively and the digits denote the numbered position of that layer among other layers of same alphabet. 88

Figure 4.3 Flow diagram of algorithm's implementation. 90

Figure 4.4 Plate 1(a) uncentered patch, 1(b) ground truth, 1(c) algorithm output demonstrating the rejection capability for pose uncentered SCN eggs; 2(a) centered patch, 2(b) ground truth and 2(c) algorithm output demonstrating selectivity of our algorithm on the same translated SCN egg. 92

Figure 4.5 Detection results with purple boxes indicating correctly labeled eggs, yellow and blue boxes show missed detections and false alarms respectively. 93

Figure 4.6 Boundary situated object (egg) scenario: (a) test frame, (b) ground truth with egg shown in a purple box, (c) missed detection (shown in yellow box) due to lack of boundary padding and (d) successful detection (shown in purple box) as a result of end correction. 94

Figure 4.7 Gray-scale images of gradual time-varying development of instability structure at two different parameter values. 101

Figure 4.8 Illustration of implicit method of generating soft labels. 102

Figure 4.9 Structure of the convolutional autoencoder with selectivity masks. The encoder portion extracts meaningful features from convolution and subsampling operations, while the decoder portion reconstructs the output into the original dimensions through deconvolution and upsampling. Best viewed on screen, in color. 104

Figure 4.10 a) Schematics of the experimental apparatus. 1 - settling chamber, 2 - inlet duct, 3 - inlet optical access module (IOAM), 4 - test section, 5 & 6 - big and small extension ducts, 7 - pressure transducers, X_s - swirler location, X_p - transducer port location, X_i - fuel injection location, (b) Visible coherent structure in grayscale images at 900 lpm AFR and full premixing for 45 lpm FFR. 107

Figure 4.11 Schematics of implementation of trained network on transition test data. . . 108

Figure 4.12 Schematics of selection of transition threshold. 110

Figure 4.13 Illustration of CSAE's ability to reproduce explicit labels. 111

Figure 4.14 Feature maps for (a) the third convolution layer, (b) the second pooling layer, (c) the fourth convolution layer, (d) the unpooling layer and (e) the deconvolution layer. 111

Figure 4.15 Code layer selection for 500_{40to30} with, (a) 8 units, (b) 10 units, (c) 30 units and (d) 40 units. 112

Figure 4.16 Results of, (a) CSAE and (b) SCRF for test transition protocol, 500_{40to30}. . 114

Figure 4.17 Results of transition protocols for, (a) 500_{40to30}, (b) 500_{40to28}, (c) $50_{700to800}$ and (d) $40_{500to600}$ where dashed arrows indicate the results for frames near the unstable flame in the transition region, and thick arrows show results for frames in the supposedly stable regions. 115

Figure 4.18 Adjacency labeling result of transition protocols for 600_{50to35} at the different regions of the profile while the image frames without any boundaries represent the inputs to the protocols at the points indicated by the arrows. . 117

Figure 5.1 Hierarchical PFSA-based feature extraction at a glance 122

Figure 5.2 Illustration of Chinese Restaurant Process with numbers: 1,2,3, $\cdots$ indicating customers' order of arrival. 124

Figure 5.3 Input-output plots of non-stationary dynamics for chaotic Duffing System under various signal to noise ratio (SNR) - Plate (a) SNR = ∞, Plate (b) SNR = 9 and Plate (c) SNR = 1. 137

Figure 5.4 Input-output plots of non-stationary dynamics for a random mix of chaotic Duffing system and Van der Pol system under various signal to noise ratio (SNR) - Plate (a) SNR = ∞, Plate (b) SNR = 1. 138

Figure 5.5 Results for noiseless Duffing system where, Plate (a) HSDF using classical CRP-only, Plate (b) HSDF using classical CRP and stickiness, and Plate (c) offline PFSA revision. 139

Figure 5.6 Results for noiseless Duffing system using on-line HSDF (i.e., without off-line PFSA revision) with classical CRP formulation under different noise contamination levels - Plate (a) SNR = ∞, Plate (b) SNR = 9 and Plate (c) SNR = 1. 140

Figure 5.7 Results for Duffing system using online HSDF (i.e., without off-line PFSA revision) with adaptive CRP formulation under different noise contamination levels - Plate (a) SNR = ∞, Plate (b) SNR = 9 and Plate (c) SNR = 1. 140

Figure 5.8 Data log-likelihood plots with $*$ representing the start of a new class for Duffing system using online HSDF (i.e., without off-line PFSA revision) with adaptive CRP formulation under different noise contamination levels - Plate (a) SNR = ∞, Plate (a) SNR = 9 and Plate (b) SNR = 1. 141

Figure 5.9 Performance of HDP-HMM approach for time series with two features under different noise contamination levels - Plate (a) SNR = ∞, and Plate (b) SNR = 1. 142

Figure 5.10 Performance comparison of (a) HSDF and (b) HDP-HMM approaches for time series with three features under no noise condition. 143

Figure 5.11 Performance comparison of (a) HSDF and (b) HDP-HMM approaches for time series with three features under SNR = 1 condition. 143

Figure 5.12 Concatenated power consumption for 3 randomly chosen end uses with homogenous characteristics in (a), and corresponding feature identification results via: (b) Online HSDF and (c) HDP-HMM; at slow time epochs each consisting of $6,172$ fast scale datapoints $\approx$ 6 hours. 146

Figure 5.13 Concatenated power consumption for 4 randomly chosen end uses with homogenous characteristics in (a), and corresponding feature identification results via: (b) Online HSDF and (c) HDP-HMM; at slow time epochs each consisting of $12,384$ datapoints $\approx$ 12-hours. 147

Figure 5.14 Concatenated power consumption for 3 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features with different ground truth labels). 148

Figure 5.15 Feature identification results of concatenated power consumption for 3 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features misclassified by both algorithms as similar while they have different ground truth labels) via: (a) Online HSDF and (b) HDP-HMM; at slow time epochs of $\approx$ 12 hours. 149

Figure 5.16 Concatenated power consumption for 4 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features with same ground truth label 3). 150

Figure 5.17 Feature identification results of concatenated power consumption for 4 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features misclassified by both algorithms as different while they have the same ground truth label 3) via: (a) Online HSDF and (b) HDP-HMM; at slow time epoch of $\approx$ 12 hours. 151

Figure 6.1 Construction of STPN: Atomic patterns (APs) and relational patterns (RPs) formulation where s represents the symbol. 157
Figure 6.2 Geographical information of wind turbines under analysis which are located in California, between 35.28-35.33N and 118.09-118.17W. 160
Figure 6.3 Representation of STPN for 12 wind turbines. 160
Figure 6.4 Discretization of a typical wind turbine systems using maximally bijective discretization (MBD). 161
Figure 6.5 Symbol sequence plot for a typical wind turbine. 162
Figure 6.6 Mutual information of relational patterns for selected pairs of wind turbines 162
Figure 6.7 Spatiotemporal pattern network for the group of wind turbines. 164
Figure 6.8 A monotonically decreasing relationship for all pairs of wind turbines when spatial distances increase. 165
Figure 6.9 Symbolic prediction of wind turbine 5 behavior with the observation of wind turbine 6. 167
Figure 6.10 Symbolic prediction of wind turbine 5 behavior with the observation of wind turbine 7. 167
Figure 6.11 MSEs in terms of symbolic prediction of wind turbine 5 power using observation from other turbines: As geographical (spatial) distance increases, MSE increases. 168
Figure 6.12 Wind power prediction for wind turbine 5 under the observation of symbol sequence of wind turbine 6 using STPN and HMM with mixture. 168

Figure 6.13 Wind speed prediction: Wind turbines 5, 6, 8, and 9 selected for demonstrating the wind speed prediction using STPN. On the left hand side, wind speed prediction of wind turbine 5 is shown using wind turbine 6. On the right hand side, prediction errors are shown for a series of turbines from east to west and from west to east, i.e., wind speed prediction of wind turbine 5 using wind turbines 6, 8, and 9 (from east to west) and wind speed prediction of wind turbine 9 using wind turbines 8, 6, and 5 (from west to east). 169

Figure 6.14 Mutual information between WBE and HVAC, WBE and LIGHTS, WBE and APPL, and WBE and MELS with the increment of time lag of 2 minutes in July, 2010. 172

Figure 6.15 STPN using variables, WBE, HVAC, LIGHTS, APPL and MELS in July. . 173

Figure 6.16 Energy prediction of HVAC, LIGHTS, APPL, and MELS in April 2010 using STPN, STPN+convex programming, FHMM, and CO separately shown in (b) for better visualization. 175

Figure 6.17 Calculated WBE from disaggregated energy values in April 2010 using STPN, STPN+convex programming, FHMM and CO. 176

Figure 6.18 Energy prediction of HVAC, LIGHTS, APPL, and MELS in July 2010 using STPN, STPN+convex programming, FHMM, and CO separately shown in (b) for better visualization. 177

Figure 6.19 Calculated WBE from disaggregated energy values in July 2010 using STPN, STPN+convex programming, FHMM and CO. 178

Figure 6.20 Energy prediction difference of HVAC, LIGHTS, APPL, and MELS in April 2010 among STPN, STPN+convex programming, FHMM and CO. 178

Figure 6.21 Energy prediction difference of HVAC, LIGHTS, APPL, and MELS in July 2010 among STPN, STPN+convex programming, FHMM and CO. 179

Figure 6.22 Disaggregation results using STPN, FHMM, and CO. The results are shown at every 5 minutes for better resolution (average value from 300 predictions in 5 minutes). 181

Figure 6.23 Disaggregation results using STPN, FHMM, and CO. The results are shown at every 5 minutes for better resolution (average value from 300 predictions in 5 minutes). 182

Figure 7.1 Volumetric dataset example of short time set of stacked frames N and overlap k with neighboring frames for stable and unstable region respectively. Note: False colors are used to aid visualization. 188

Figure 7.2 3D Convolutional selective autoencoder framework that accepts bursts of volumetric frames for exploring the short ranged relationship between 2D frames in a windowed form and spatial relationship between adjacent volumes, and activating volumetric features' examples in training for each region. Dropout=0.3 in corresponding dropout layers (Asterisks (*)). 191

Figure 7.3 Figures showing the performances on a) the training dataset, of b) 3D-CSAE and b) 2D-CSAE with all frames evaluated in the KL-divergence metric, and their respective thresholds. 196

Figure 7.4 Evaluation of the detection results and the visualization of volumetric frames for protocol 500_{40to30}. Arrows show direction of optical stream flow. 197

Figure 7.5 Evaluation of the detection results and the visualization of volumetric frames for protocol 600_{50to35}. Arrows show direction of optical stream flow. 198

Figure 7.6 Evaluation of the detection results and the visualization of volumetric frames for protocol $500to600_{40}$. Arrows show direction of optical stream flow. . . . 199

Figure 7.7 Evaluation of the detection results for protocol 500_{40to30} showing the first, identified and final frames of the specially highlighted points. 200

Figure 7.8 Evaluation of the detection results for protocol 600_{50to35} showing the first, identified and final frames of the specially highlighted points. 201

Figure 7.9 Evaluation of the detection results for protocol $500to600_{40}$ showing the first, identified and final frames of the specially highlighted points. 202

Figure A.1 Steps in CUDA computings 210

Figure A.2 OpenMP extension language schematic. 211

Figure A.3 Schematics of the CUDA processing flow with GeForce 8800 illustrative GPU. 213

Figure A.4 GPU Architecture Overview showing the instruction multiprocessing threads. 214

Figure A.5 Fermi streaming microprocessor CUDA core with FP floating point and INT integer computed in different streams. 215

ACKNOWLEDGMENTS

I would like to take this opportunity to express my profound gratitude to those who helped me in the various, but immense ways while conducting this research. First and foremost, Prof. Soumik Sarkar and Prof. Baskar Ganapathysubramanian for their guidance, patience and support throughout the period of this research and the writing of this thesis. Their insights and ideas have often inspired me, and renewed my hopes for completing this phase of my graduate studies. I would also like to thank my committee members for their efforts and contributions to this work: Prof. Namrata Vaswani, Prof. Asheesh K. Singh and Prof. Theodore Heindel for carefully reading my research reports and providing useful comments on how to make necessary improvements. I would like to acknowledge the many sponsors of, and key contributors to, my research and doctorate program. Some of the research works were supported by the Iowa Soybean Association through the research of Dr. Arti Singh, the Regents Innovation Funding and Rockwell Collins Inc., as well as NVIDIA® for donating the graphics cards that facilitated the quick deliveries on the research projects. I would like to thank the National Science Foundation for supporting many of the projects. I would like to acknowledge the contributions of my past and current lab mates in Self-aware Complex Systems Laboratory to the success of my research. The words of encouragement from Mr. Kin Gwn Lore, Mr. Zhanhong Jiang and Dr. Chao Liu, who were with me from the start when it was tough and rough, cannot go unmentioned. I am also thankful to have an undergraduate honors student, Paige Katherine Boor, who improved my efficiency during the last phase of the research. I am eternally grateful to Mrs. Debbie Younkin for always willing to carefully find the missing parts of my writings. Beyond that, the love shown to me by the Younkins' family and the members of international family at Stonebrook church have been the catalyst for my personal wellness during the program.

ABSTRACT

With the advent of ubiquitous sensing, robust communication and advanced computation, data-driven modeling is increasingly becoming popular for many engineering problems. Eliminating difficulties of physics-based modeling, avoiding simplifying assumptions and ad hoc empirical models are significant among many advantages of data-driven approaches, especially for large-scale complex systems. While classical statistics and signal processing algorithms have been widely used by the engineering community, advanced machine learning techniques have not been sufficiently explored in this regard. This study summarizes various categories of machine learning tools that have been applied or may be a candidate for addressing engineering problems. While there are increasing number of machine learning algorithms, the main steps involved in applying such techniques to the problems consist in: data collection and pre-processing, feature extraction, model training and inference for decision-making. To support decision-making processes in many applications, hierarchical feature extraction is key. Among various feature extraction principles, recent studies emphasize hierarchical approaches of extracting salient features that is carried out at multiple abstraction levels from data. In this context, the focus of the dissertation is towards developing hierarchical feature extraction algorithms within the framework of machine learning in order to solve challenging cyber-physical problems in various domains such as electromechanical systems and agricultural systems. Furthermore, the feature extraction techniques are described using the spatial, temporal and spatiotemporal data types collected from the systems. The wide applicability of such features in solving some selected real-life domain problems are demonstrated throughout this study.

CHAPTER 1. OVERVIEW

The success stories recorded on real world applications by machine and deep learning algorithms are abundant, especially in many sectors including academic, business, Internet, communication and health care industries. Tasks such as recognition of handwritten digit (LeCun et al. (1998a)) have been useful for identifying authenticity of signatures on checks (LeCun and Bengio (1995)), while processing of speech and natural languages with deep convolutional networks (Collobert and Weston (2008)) have found uses in audio and video-based tasks.

In the health-care sector, machine learning algorithms are embedded in devices to diagnose heart failure, assimilate seizure patterns and to detect growth of tumor cells among many other applications in health diagnosis and drug delivery (Volk et al. (2012)). Similarly, improved security techniques such as employer access control by Amazon and cognitive computing for medical applications and discovery by IBM all employ one form of machine learning techniques or the other. Some Internet applications for everyday convenience purposes include identifying spam emails at Google Inc., product recommendation by Amazon, movie and music recommender systems by Netflix Inc. and Spotify respectively, graph search at Facebook Inc., with the list becoming endless by each new day. All of such algorithms in use to achieve these goals of making lives convenient underscore the applications of large-scale, data-driven optimization. While these and other similar applications are having transformative impact on society, the spot-light is on many possible core engineering and agricultural tasks that could benefit from machine and deep learning tools. In that regard, this research attempted to bridge some gaps by exploring the potential real-life benefits of feature extraction, using machine and deep learning tools to successfully solve some engineering and plant science problems.

1.1 Introduction

A major on-going research in the area of machine learning and artificial intelligence is the aspect of extracting salient features from data. Feature extraction is an important initial step to any machine learning task (after data curation) for making efficient, machine learning algorithms. For instance, a query task would require gathering only the information related to the query at hand through some form of expert knowledge or models which are able to discern certain patterns from data. Among several possible engineering applications, this research focuses on applying novel feature extraction methods for engineering problems (e.g., early detection of combustion instabilities) and agricultural problems (e.g., plant stress phenotyping for improving yields). While these problems and others to be discussed represent very different physical systems, the research will attempt to show the effectiveness of similar concept of hierarchical feature extraction formulations from different perspective, based on best current practices that are able to extract the 'best' features. The main elements of the hypothesis that this research will enhance are described by the following 'what' questions. What datasets best explain the interesting features related to the problems? How quick are such datasets available for analysis? How can one quantify the information content of such dataset? What volume is required to extract the right features from the datasets? How does one find out what information are irrelevant, redundant or constitute undesirable properties? These kinds of questions are still open-ended because they depend largely on the problem at hand. In the context of our applications, these questions have huge impacts where there is a need to embed the available domain/expert knowledge about the system within the feature extraction process. In addition to the above, further questions to be answered in the application of feature extraction to agricultural systems are related to both 'how' in addition to the 'what' as follows. How can we improve the efficiency of management practices with the automated phenotyping results (Domingos (2012))[1]? What are the yield potentials per stated size of available resources? How can the yield losses be reduced? How can farmers understand what varieties of their crops produce the most yield,

[1] Just as farmers combine seeds with nutrients for crops to grow, engineers and data scientists combine knowledge with data to grow programs!

or what areas of their farm are the least affected by pests and diseases? In order to improve yield, what information could be provided to consultants and researchers for developing better cultivars? The last set of questions are related to the algorithm development and tool design that will enable automating the feature extraction process and its applications. The design and development are largely divided into two parts, namely, the software and hardware. The software part has such questions as follows. What back-end platforms are available and efficient in computation? What front-end user interface are easy to develop and use for a given application? Lastly, but by no means the least, a subtle question is: what drivers are needed to communicate software information to the hardware? For the hardware part, designers (importantly also, users) are interested in knowing the most friendly device support are available.

1.2 Research Goal

The main goal of the research is to extract hierarchical features from the high dimensional datasets that are characteristics of physical systems in general. Such high dimensional datasets can be considered from spatial (i.e., those changing with space) to temporal (i.e., those changing with time) and the more commonly encountered spatiotemporal (i.e., those changing with both space and time). The models for approaching the kinds of observations are usually non-trivial. However, the appeal for them are in their non-obstructiveness of the working of the systems. For automation, most problems of these kinds are complex and would require some level of intelligence built into a machine (called algorithms) to recognize the presence of signatures of patterns; for instance, subclasses of stress types on individual plants or the lower dimensional representation of coherent structures. In these applications, the large amount of high dimensional, multi-scale example images can only be mapped into low dimensions by complex mathematical relationships learning many parameters. With such complexity, the functions are usually not analytically tractable in most cases (Sipser (2013)). Therefore, the entire learning pipeline requires efficient algorithms and software architecture to acquire, save large amount of data to gain adequate insight into several aspects simultaneously. This is the main motivation of the present research work whose goal is

to design machine learning for hierarchical feature extraction from a large variety of engineering and plant-related datasets for intelligent decision-making processes. Hierarchical feature extraction can be described as a process through which application-driven features, and features-of-features are learned and extracted from data at multiple logical layers or levels of abstraction. The study will show different techniques that have been developed for the extraction process, as well as their shortcomings and improvements made. Due to the diversity in the application areas of the study, it is motivated from the perspective of data characteristics introduced earlier as spatial, temporal and spatiotemporal types. This indeed makes sense since the patterns to be extracted are unique to them. Some of the specific applications in electromechanical and cyber-agricultural systems that are investigated in this research are used to motivate the goal in Subsections 1.2.1, 1.2.2 and 1.2.3.

1.2.1 Spatial data applications

The sensed datasets in the engineering and cyber-agricultural systems considered are mostly vision-based, thus spatial in nature. It may be argued that this categorization arises from a simplification of the real observations, or is based on the kind of study pursued. Working based off that simplification, plant biologists, breeders and agronomists for instance have a dire need to assimilate and analyze rapidly evolving, enormous amounts of data collected from the field by farmers, which are aided by improvements in aerial and on-site data capturing devices (Mitka and Bart (2015); Li et al. (2014)).

In military and commercial applications, capturing good quality images and videos are key in numerous critical decision-making tasks, ranging from security applications, path planning for unmanned vehicles to medical diagnostics and commercial recommender systems. Large scale applications along with cost constraints often limit capturing high quality images/videos due to corruption. Furthermore, adverse lighting conditions, such as low-light, night time, very bright light, dusty/foggy/cluttery environments worsen the situation. Figure 1.1 shows an example of a badly degraded environment which may pose enormous difficulty for cheap, low dynamic range sensors. Therefore, the area of image/video denoising has seen an abundance of recent interest from both

Figure 1.1: Example of a Degraded Visual Environment (DVE) - dusty condition ("courtesy: www.ensco.com").

academia and industry. Application domains, such as Intelligence, Surveillance and Reconnaissance (ISiniR) missions (e.g., recognizing and distinguishing of enemy warship from a large distance), unmanned vehicles (e.g., automated landing zone selection for UAVs or self-driving cars), medical diagnostics (e.g., brain Magnetic Resonance Imaging (MRI)) and precision agriculture (e.g., early detection of plant defects using visible discoloration) are some of the motivation of our approaches. Such problems are approached via contrast enhancement and image denoising models that are trained to learn underlying signal features in low-light images.

1.2.2 Temporal data applications

Data types varying with individual single dimensional independent variable are termed temporal datasets. The dependent variables are traditionally a function of time, being times series. Nowadays, time series encompass signals that change with respect to any quantity of interest. Speech recognition datasets, such as the speaker diarization data (Tranter and Reynolds (2006)) and farm yield prediction using physical markers are among the standard examples of dataset considered in this category of observations. A recent blog (Picheny (2016)) by IBM Watson developers, titled 'Look whos talking: IBM debuts Watson Speech-To-Text Speaker Diarization beta' describes the industrial relevance placed on the application from the speaker diarization dataset example. An illustration is shown in Figure 1.2 of a businesswoman having a meeting with her business associates and partners. The speech signal of each participant varies with several factors such as length of speech and frequency of taking turns. Therefore, the dataset is high dimensional. However, with the speaker diarization problem, given an observation from such a meeting where there is no knowledge *a priori* of the number of participants, and the time period of speaking allotted to each participant. It is of interest to determine the number of participants and label each participant's interval of speaking from the speech pattern records. This idea is usually to extract the most interesting low dimensional features that embed effectively the high dimensional datasets.

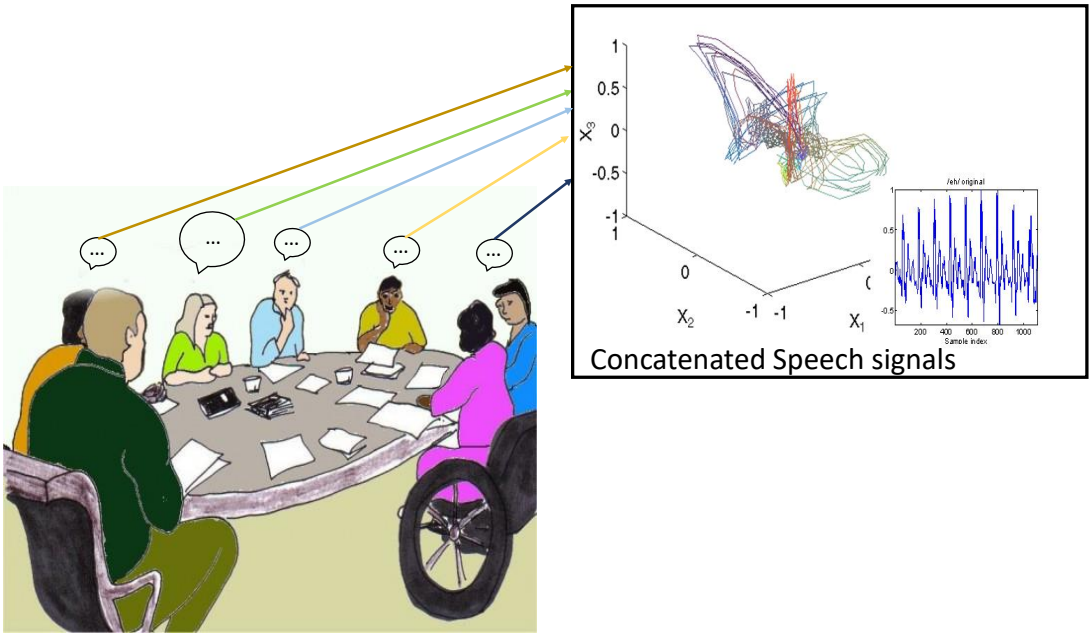


Figure 1.2: High dimensional speech signals of a set of business people in a meeting: Courtesy of "Cartoon resources on Google".

1.2.3 Spatiotemporal data applications

High dimensional datasets varying in both space and time are termed spatiotemporal datasets. Since most events in real life are time and space-based, these signals are the most common type of signals available. To motivate the techniques for addressing these data, we introduce applications such as the problem of early detection of instability characteristics in the combustion chambers of dynamic systems. Such problems are key for anticipatory monitoring and safety-driven actions in the operations of the engines. The patterns can be considered spatiotemporal because the systems that generates the observations from which instabilities are detected are dynamic in nature, and such dynamics introduce the temporal dimensions to the already spatial location. In particular, strict NOx and other particulate gas emission laws have resulted in a shift of large chunk of engines design issues to the domain of engineering control, especially in power generation systems and jet propulsion engines (Samad and Annaswamy (2011)).

With that, the ensuing approach to mitigate the engine-related pollution hazards has been in favor of leaner (i.e., increased air-fuel ratio), premixed (controlled by the upstream distance from the combustion chamber where the air and fuel mixture takes place) equivalent mixture that is atomized in the combustion chambers (Huang and Yang (2009)). Consequently, a system burning the mixture normally, i.e., in a stable state experiences some shorter time (frame) scale structures, where it transitions to an unstable state, resulting in blowout at those time scales. The main problems with such design are therefore the instabilities which usually become more prominently sustained at longer time-scales. The hypothesis that was examined in this research was how to enhance the detection of the instability signatures from flame images, pressure data, chemiluminescence recordings, etc., using the novel cutting edge approaches introduced. Again, the idea depends on extracting the exact patterns that represent the objects of interests.

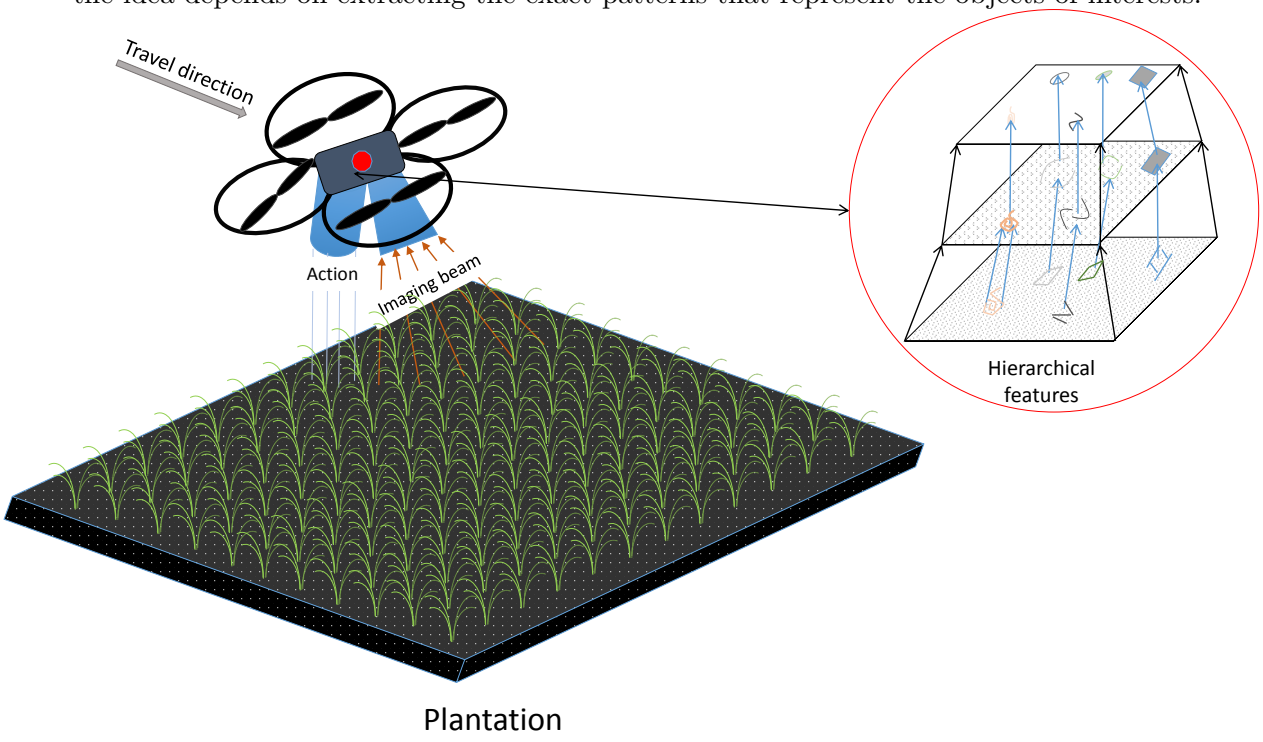


Figure 1.3: Automated aerial imagery for pattern recognition and efficient crop management.

In most real life cases, there are combinations of interacting systems, such as where one system generates and another system analyzes the generated datasets. Figure 1.3 is a schematic that

illustrates a quad-copter with on-board sensors, hovering over a field to detect the presence of undesired substances. The embedded algorithm is made intelligent to perform the computations such as calibrating the severity level of rodents or disease infections to come up with the most appropriate action that is performed by a different module. It specifically illustrates an electromechanical control system which is equipped with fast data processors for on-board detection of hierarchical features from soil particles. In this regards, some of the extracted features may indicate the presence of harmful microorganisms such as pests or the absence of certain soil water or fertilizer conditions. As a result of such real-time on-board extractor actions (e.g., applications of pesticides, water or fertilizers) are taken.

Energy prediction problems on the other hand are essential for operating, monitoring, and optimizing (in terms of efficiency and cost) diverse energy systems, from the supply side (e.g., wind energy, solar energy, power systems, storage) to the demand side (e.g., load monitoring, usage of electric vehicles, building energy management). Numerous studies that are of significance to organizations such as National Renewable Energy Laboratory (NREL) are being carried out in terms of predicting energy generation/consumption using time-series data (Ziel et al. (2016); Liu et al. (2015); Alessandrini et al. (2015); Zuluaga et al. (2015); Wang et al. (2015); Garshasbi et al. (2016)).

1.3 Contributions

The objective of this research was to develop a framework that is suitable for data preprocessing and features extraction from datasets following the idea of determining what constitutes rich descriptors in respective applications. These broad topics of preprocessing and feature extraction will be considered based on the applications in a consistent manner. The over-arching aims were to explore the benefits of current best computational science algorithms, high performance computing and software platforms for feature extraction from datasets by reducing (or sometimes expanding) data and feature space dimensionality to one where the algorithm is capable of learning the 'best' task-based features. The contributions are divided into 2 categories, namely, the theoretical and

practical algorithmic contributions. The main theoretical contributions of this research are itemized as follows.

- Knowledge improvement by contributing meaningfully to the development of algorithmic frameworks for extracting rich, robust and reliable descriptors (features) from spatial, temporal and spatiotemporal data.
- Architectures review for optimally combining the heterogeneous features in a scalable way to handle multiple applications (Mitchell (2006)).
- Algorithms' computational efficiency optimization as measured by the speed, accuracy and memory requirement to enhance the efficiencies of the feature extractors.
- Identification of suitable test datasets with all the typical characteristic of 'big data',
- Seamless interfacing of architectures with the aid of transparent graphics and central processing units (GPUs and CPUs) computation.
- Analyses and discussions of results obtained from the applications of the algorithms to real-world data in the light of some of the hypotheses.

Some of the algorithm-based contributions of this research are itemized as follows:

- Development of a Low Light Network (LLNet) to extract brightening features from synthetic images for enhancing naturally corrupted images taken in noisy/corrupt environments or by faulty sensors.
- Development of a streaming-type hierarchical feature extraction algorithm for energy disaggregation from standard reference energy disaggregation data sets.
- Automated segmentation and counting of soybean cyst nematode eggs present on microscopic image frames for worm management and control.
- Automated detection of subtle features of coherent structures from hi-speed flame videos that indicates the commencement of combustion instabilities.

1.4 Organization of Dissertation

The dissertation is approached from the perspective of feature extraction from the three main datasets. The Chapters 1 (the current one) and 2, present brief overviews of the techniques used throughout the dissertation. Chapter 3 to Chapter 7 are dedicated to extracting features from the datasets based on the fundamentals presented in Chapter 2. Specifically,

- Chapter 2 presents a succinct review of the available algorithm and methods that has culminated into the current state-of-the-art deep learning technique for feature extraction.
- In Chapter 3, hierarchical features that characterize basic 2-dimensional images are extracted via a low light network in order to facilitate transfer learning for image enhancement and denoising.
- In Chapter 4, a novel convolutional selective autoencoder approach is proposed for object identification in both cluttered images as well as unsupervised learning of features from spatial dataset.
- In Chapter 5, a novel hierarchical symbolic dynamic filtering technique is proposed for extracting features of features from temporal datasets – publicly available Reference Energy Disaggregation Dataset (REDD).
- In Chapter 6, a spatiotemporal pattern network (STPN) is enhanced by constrained optimization for analyzing energy demand by individual household appliances from the whole building dataset and also, the energy supplied from an array of collocated wind turbines.
- In Chapter 7, a 3D-version of convolutional selective autoencoder is developed for spatiotemporal characterization of the scales and onset of intermittent structures.
- The dissertation is summarized and concluded in Chapter 8. Also, a few recommendations of future research directions that are either on-going or possibly feasible in the light of this research's achievement will be described.

- Appendix A presents an introduction to the description of the architectural platform and hardware used for the research work.
- In Appendix B, the most commonly used software for algorithm implementation are compared and evaluated.

CHAPTER 2. REVIEW OF LITERATURE

In this chapter, the available wealth of knowledge that builds up to developing the deep learning techniques are carefully reviewed. The basic curve fitting algorithms – linear to polynomial techniques – in regression methods to statistical Bayesian inference algorithms, undirected graph-based methods, basis functions are kept in the background; to develop powerful models that are able to capture the low dimensional manifold features that are descriptive of a given complex dataset.

2.1 Introduction

By design, machine learning algorithms incorporate statistical reasoning, computational intelligence, information theory, decision theory and optimization theory; making them suitable for learning without explicit programming. In other words, machine learning algorithms help computers to observe the world, learn from it and make intelligent generalizations of it. The word machine learning is known to have been coined by Arthur Samuel in 1950 for algorithms that are capable of modeling intelligent human behaviors. The major elements of machine learning are the data preprocessing, transformation and feature learning. Problems in the purview of machine learning are treated as pattern recognition problems (Bishop (2006)) in engineering applications while psychologists and neuroscientists (Mitchell (2006)) have similar questions related to statistical learning. From an engineering standpoint, adequate study of pattern recognition and task performance would constitute, but is not limited to: the major categorization of learning and modeling, the steps involved in recognizing the patterns and finally the hierarchical feature extraction concept in the data domains. The hierarchy of steps in this machine learning review for feature extraction is depicted in Figure 2.1.

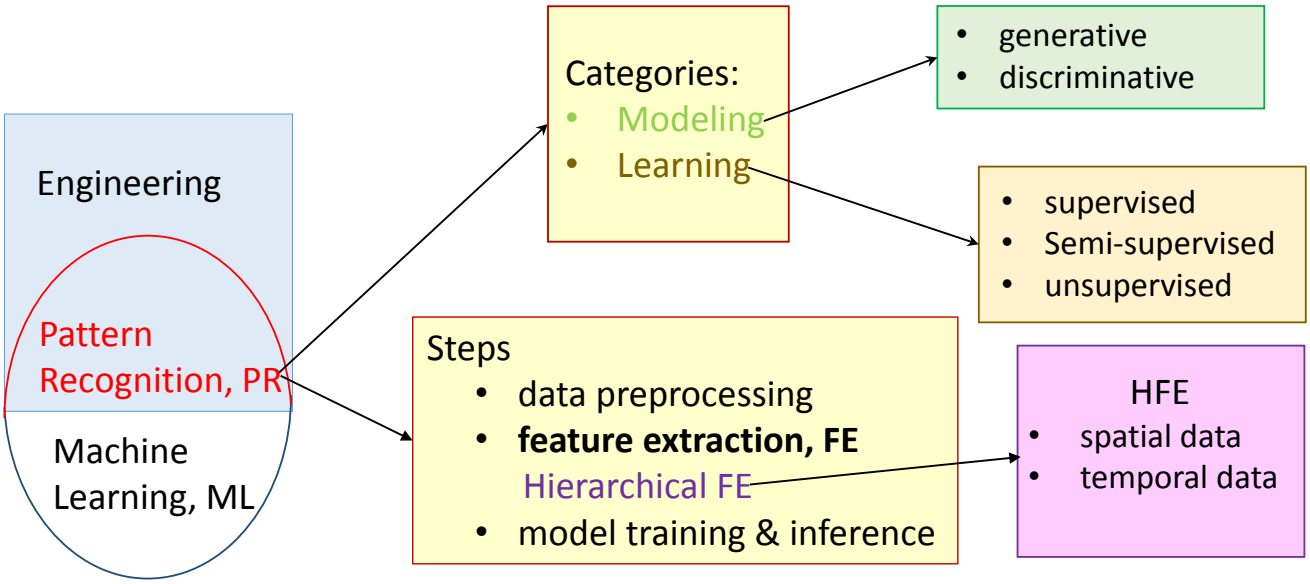


Figure 2.1: Machine learning applied to engineering problems with a focus on hierarchical feature extraction.

2.1.1 Learning model - REO

A useful way to think of learning is, as a series of systematic operations of Representation, Evaluation and Optimization (REO), resulting in the model that was described by (Domingos (2012)). By representation, the authors were referring to a set of hypothesis space that the particular data must fulfill for a chosen classifier to be effective. The authors further described the need to properly fit the data to the right classifier, i.e., one that is devoid of over-fitting or under-fitting. In the evaluation stage, the simplest[1] objective function that models the data is designed to be optimized by solution space search methods that determine how well the classifier achieves the objective set out in the evaluation function. In order to develop each individual components of the model, several desiderata and nuances that could guide beginners to designing machine learning tools were introduced. Some of the major factors to consider in selecting a specific algorithm are: bias/variance, linear fit/decision tree models, beam/greedy search algorithms, training problems that includes curse of dimensionality, blessings of uniformity, etc., among others.

[1] "As a rule of thumb, a dumb algorithm with lots and lots of data beats a clever one with modest amounts of it" (Domingos (2012))

2.1.2 Learning model - TPE

Perhaps, another perspective for describing the ideas that make up machine learning were provided by (Mitchell (2006)) using a TPE model that is described by the performance, P of an algorithm due to its experience, E on a specified task, T. Some of the experiences to be garnered by the algorithm could be in the form of training dataset while those randomly set aside from the training set, called held out sets, and others, possibly from the same field of application differently from the training enable inference. Attempts at streamlining the categorization based on the TPE model would enable it to be done in a concise manner. For instance, in the study (Singh et al. (2015)), plant scientists and engineers categorized machine learning algorithms which have real and potential applicability for the new paradigm of efficient, non-destructive type phenotyping of stress types in plants. The authors proposed all the applications to belong to any (or combination) of the Identification, Classification, Quantification and Prediction, ICQP model in the practice of phenotyping stress types in plant (Mitka and Bart (2015); Li et al. (2014)). Furthermore, there are other functions that have been reported such as information retrieval (Madsen et al. (2005)) and natural language processing (Collobert and Weston (2008)).

2.2 Learning and Modeling Paradigms

There are several ways to categorize the large number of machine learning algorithms as introduced in Subsection 2.1.2. While the ICQP may be a T-task based classification under the former, there are a more wider classes of models under the task categorization. The ICQP-model (Singh et al. (2015)) for example was a categorization of the major tasks that are needed to achieve the goal of stress phenotyping in plants. Recently, deep learning-based approaches gained immense traction as they have been shown to outperform all other state-of-the-art machine learning tools for a large variety of computer vision applications such as object recognition (Krizhevsky et al. (2012a)), scene understanding (Couprie et al. (2013)) and occlusion detection (Sarkar et al. (2015d)). Therefore, in this study, the applicable learning categories – supervised, unsupervised and semi-supervised

classes are reviewed. Also, models have been approached from the perspective of how the outputs are derived from the input dataset.

2.2.1 Supervised, semi-supervised and unsupervised learning

In supervised learning, a function (not necessarily analytic), $f : X \rightarrow Y$ relates the input, X to some known output labels, Y corresponding to each input value (Bishop (2006)). In this class belongs a regression problem for learning parameters relating the yield (i.e., labels) to a set of influencing external and specie-related conditions (inputs). However, the appeal for such learnt parameters, devoid of over-fitting, lies in their ability to predict labels from a different combination of input factors. Usually, these new combinations are a random different combination within the sphere of its learning. The major setback in this form of learning is the high cost associated with labeling for each new task. On the other end of the spectrum are the segmentation techniques such as marker type watershed models, hierarchical and partitional clustering (Wang (2010)), cross-modal clustering (Coen (2005)) that have similarity to hard and soft thresholding. These models constitute the group of unsupervised learning. Algorithms in this class are mostly suited for anomaly detection type for example discoloration on surfaces of leaves may be suited for these classes. One of the key innovations that came out of the deep learning community is learning hierarchical features in an unsupervised manner with deep Boltzmann machines (Salakhutdinov and Hinton (2009, 2012)). In the nonparametric modeling community, similar problems are taking a center stage for considerations. For example, Hierarchical Dirichlet Process over Hidden Markov Models (HDP-HMM) have been shown to be efficient for automatic speaker diarisation problems (Fox et al. (2011)), where 'who spoke when' (Tranter and Reynolds (2006)) have to be identified from a audio time-series without knowing how many speakers are present. Modeling of high-dimensional data was explored in (Pitsikalis et al. (2003)) for analysis of speech data through generalized fractal dimensions and Lyapunov exponents modulated by adjusting two hyper-parameters of time delay and embedding dimension. In addition to hierarchical extraction of features, low memory, online type applications were considered (Akintayo and Sarkar (2017)). In general, many machine

learning models can also be learnt in a semi-supervised manner. With reference to biologically-inspiration, an overcomplete dictionary of activities is learnt by animals at a given point of training instance. The presence of redundancy learnt from a given application helps to supervise a related task. In computational modeling for applications, overcompleteness added to sparsity (Elad and Aharon (2006)) are important spices that enhance the functionality of algorithms. Artificial neural networks for instance are an example of how algorithms learn from a training data – gaining experience – to apply on different test sets. Convolutional neural networks (LeCun et al. (1998a); Collobert and Weston (2008)) are among the best known and preforming examples of the neural network. The architecture learns joint weights from invariant features in examples for effective local neighborhood characterization. Energy-based Boltzmann machines (Salakhutdinov and Hinton (2009, 2012); Bengio (2009); Ranzato et al. (2007)), information theoretic and manifold learning inspired auto-encoders (Vincent et al. (2008)) are other architectures that learn features from labeled data (Erhan et al. (2010)) for inference on different tasks.

2.2.2 Generative and discriminative models

A way to classify learning models is to consider how the outputs are estimated from input data. In some cases, it may be desirable and effective to learn the parameters of a joint distribution, $P(X = x, Y = y)$ of the input and labels for a supervised learning or hidden units for an unsupervised learning while other circumstances favor learning the parameters of a conditional distribution, $P(Y = y|X = x)$ from the input. The former classes are the generative models and the latter fall under the discriminative models. The classical example-pair by authors (Mitchell (2006); Ng and Jordan (2001)) are the linear classifiers and naive Bayes', its analog logistic regression respectively for each group. When data is sparse and the model is more reliable, the generative model are more suitable since the increasing error soon becomes asymptotic while the paradigm of enormous amount of data favors asymptotic error reduction (Ng and Jordan (2001)) obtained in discriminative models. The goal of generative models is to sample data from simpler (easy to sample) distributions that generalize the finite data samples. Some of its advantages are in pre-

diction, uncertainty estimation, missing data imputation, model selection and sample generation. However, discriminative models are favored because of the tractability of the training components, low computational requirements (Erdogan (2010)) and their stability in training. Discriminative models clearly require lower number of parameters to be estimated than the generative models, but with the disadvantage that they have to be well regularized than those of the generative model.

2.3 Machine Learning Procedure

Given the learning and modeling categories (Section 2.2), the main steps followed by a typical machine learning tool-chain are outlined and discussed in the following subheadings: data generation and preprocessing, feature extraction and model training and inference.

2.3.1 Data pre-processing

A first step in many learning activity is based on knowledge of the domain of data because features to an application might be disturbances to other applications. In a data-driven framework, the features underlying many processes can best be the data that they are derived from assuming the process is devoid of external disturbances. However, the authors in (Smola and Vishawanathan (2008)) notes that new challenges and problems are made easier by adaptation of techniques when they possess similar data types and/or exhibit similar relations among variables to a previously tackled problem. However, some form of pre-processing are also important for removing unwanted artifact like noise from data. The quality of datasets from applications have to be examined closely. Certain qualities such as accuracy, completeness, consistency, interpretability and trust in the source have to be verified. Based on the required qualities, tasks can be examined under the following categories: cleaning to tackle incompleteness, inconsistencies in units, values, and ensuring robustness to high frequency contents. Other processes that are included in preprocessing are data integration, transformation, reduction and discretization. Improvements in image enhancement tasks are specifically important as they serve as useful preprocessing tools to aid further analysis. The authors of (Vincent et al. (2008)) showed the concept of denoising auto-encoders for automat-

ing noise removal from images while (Jain and Seung (2008)) applied convolutional neural networks for the same purpose on natural images. Enhancement procedures such as inpainting (Xie et al. (2012)) and deblurring (Schuler et al. (2014)) that have been reported in neural network frameworks are some other techniques that applies learning to pre-process. An adaptive multi-column architecture was implemented by Agostinelli et al. (2013) to robustly denoise images with varieties of synthetically added noise types. Stacked denoising auto-encoder was used (Burger et al. (2012)) to reconstruct clean images from noisy images by exploiting the encoding layer of the multilayer perceptron (MLP). Apart from being a preprocessing step for learning algorithms, this step has tremendous engineering benefits for industrial and military applications. For instance, robotics vision, drones imaging, military airplanes imaging sensors can be aided by adequately preprocessed data. A recent study conducted by the present author with his team (Akintayo et al. (2015)) has grouped the sources of low-light "corruption" in sensed images into three major categories. They are based on the effects of: the environment described as degraded visual environment that arises from unfavorable weather conditions such as fog, snow, etc., and high dynamic range of the imaging scene; the sensor effects due to size, weight and power cost that limit the extent of improvements to sensor devices; the image processing and display effects of discretization, image de-mosaicing (deriving the channels for colored images), and smoothing effects in greyscale images. A sparse denoising auto-encoder framework, called low light network, LLNet (Lore et al. (2017)) was explored to enhance images taken with poor sensors and/or in degraded environmental conditions due to its reviewed benefits and track record. Also, another framework, ReProcCS (Guo et al. (2014)) which does effectively background and foreground separation was used to detect objects of interest in a dark scenario. All the above pre-processing goals however, the basic techniques for most algorithm are standardization, normalization and in some cases, pre-whitening.

2.3.2 Feature extraction

An important aspect of learning is the discovery of salient features that clearly characterizes the goal to be achieved - feature extraction. Due to its importance, several machine learning and image

processing programs have dedicated toolkits and libraries such as Python language's `scikit-learn` that are dedicated solely for feature extraction. Features – also called parameters, features vectors, code vectors, descriptors – are usually unique attributes that describe a process. For instance, the health condition of an individual has different effects, the features could be thought of as the underlying symptoms. An important property of these features is adequacy. In that perspective, the features must just exactly represent the underlying process by being robust to all other high frequency contents. It is the satisfaction of these properties that authors (Bengio (2009)) called good representation. In some neural network algorithms, for instance, deep belief networks, (Bengio et al. (2007)) Restricted Boltzmann Machines (Hinton and Salakhutdinov (2006)) and autoencoders (Vincent et al. (2008)), feature extraction step is called a pre-training stage. Feature extraction also falls into the category of data mining and knowledge discovery. It is an attempt to unravel the "unknown" feature from data with machine learning algorithms. However, the word "unknown" here is related to the algorithm's discovery, automatic feature construction since the features are usually known, *a priori* to the domain experts. Feature construction is then followed by efficient search through a reduced set of features and then computing a criterion for assessing which features are the most descriptive of the process. Therefore, the features are a lower dimensional abstraction of the dataset. In this manner, data is projected to a lower dimensions manifold (Vincent et al. (2008)), thus relating feature extraction to dimensionality reduction. The primitive algorithms are the Fisher's linear discriminant and nearest neighbor algorithms. The discussion was developed to the "state-of-the-art" feature extraction methods exemplified by support vector machines, multi-layer perceptrons and ensemble methods. Thus, features can be imagined as classifier's prior knowledge of an underlying application that aids it to perform the desired task.

2.3.3 Model training & inference

The fidelity of the extracted features from the model learning stage are evaluated in the training stage on different tasks. It is basically a feedback stage where the performance, in TPE model (Mitchell (2006)), of applying the learned features are evaluated and the error are fed back into the

learning algorithm until a threshold in error is achieved. For example, neural networks are typically trained using the so-called "backpropagation" algorithms that use the error feedback to improve the models. Previously, training of layer-wise learned network in this manner had proven difficult until (Bengio et al. (2007)) introduced techniques for achieving it in a "greedy fashion". The training process typically includes a regularization function as in the (LeCun et al. (1998a)) without which the error profile would not generally be monotonically decreasing and the model becomes prone to over-fitting. During optimizing the model parameters, the momentum-based parameters for example, Nesterov-momentum (Sutskever et al. (2013)) are used to adaptively modify the step sizes in optimization algorithm to reduce the tendency of getting stuck in local minima. Machine learning algorithms other than neural networks have different training schemes. Usually, this is the aspect that differentiates the various machine learning algorithms available. For instance, (Erdogan (2010)) notes that least square regression method and a fisher linear discriminant (FLD) only differ by the presence of a regularization factor in the former's optimization, training or loss function. The trained model is finally used to perform similar task on a new test data which was either held out of the training set also "out-of-sample" or an entirely new dataset. This final step is called the inference step as statistical decisions are made with the trained models.

2.4 Hierarchical Feature Extraction (HFE)

Among the various feature hierarchical feature extraction technique has been found to perform better because of its human-like inference making property. The problem of hierarchical feature extraction is key to the success of many applications such as robotics, complex system modeling and image processing. For autonomous perception issues in robotics applications, environmental features are extracted in a hierarchical manner where lower level features may signify objects in the scene and higher level features represent contextual information needed for planning (Lai et al. (2014)). Similarly, hierarchical feature extraction in complex systems falls under the category of switched and hybrid system modeling approaches (Duarte Antunes (2013)). Recent successes of deep learning in image, video and speech processing applications show the efficacy of hierarchical

feature extraction using machine learning (Hinton and Salakhutdinov (2006); Bengio and Olivier (2011)). In addition to the successes recorded with such structures, cognitive processes in humans also show that ideas are generated in an adaptive and hierarchical manner. This conjecture has provided, and continues to provide, a widespread intuition to modeling human learning and reasoning processes using probabilistic programming concepts (Tenenbaum et al. (2011)). Hierarchical feature extraction in this study is classified based on the data category to be analyzed as spatial, temporal or spatiotemporal. Given the appropriate techniques and suitably extracted features, the major benefit of feature extraction is in making the best informed decision about the underlying process or system.

2.4.1 Spatial data

Data types of points, lines, regions and their combinations are the bases of spatial data analysis (Erwig et al. (1999)). The geometries are mainly produced by imaging devices such as camera and scanners, and have been widely used in such applications as medical imaging. They are purely location-based data useful for visualizing shapes, colors and spreads, but can have all the artifacts present in other classes of data. The data class usually embed rich features because of their high dimensionality, thus they require automated feature extraction techniques. Its disadvantages are however that: its volume which can get large, thus requiring large space in databases; absence of dynamics which limits the knowledge of how the system behavior is changing. Deep networks have been particularly useful for these classes of data. In scene labeling applications for instance, feature extraction was considered a major part of steps in understanding images (Farabet et al. (2013)). The authors proposed using a deep learning architecture – convolutional multi-scale, dense feature extraction from pixels that are embedded in multiple locations around a certain pixel. The authors also discussed the advantages of multiple post-processing to achieve fast labeling of images. Arriving at good representation verified by contextual consistency check was important with their hierarchical feature extraction being enhanced to end-to-end training of the network. For graph-based classification, input images were transformed using a Laplacian pyramid, then a pixel-wise

classifier to group the features followed by over-segmentation techniques done on super-pixels, conditional random field (CRF) over super-pixels and multilevel cut with class purity criterion – comparison. It is noted that most contexts are global in nature, and they would therefore pose a difficulty of making local decisions from global context. Most of the applications of our algorithms have been tailored to extracting and smartly modifying features that emanate from spatial data analysis.

2.4.2 Temporal data

Temporal dataset have been introduced as the dataset types that are traditionally called time series. The series or signals are mostly single dimensions, only produced at discrete (bursts) or continuous time intervals. They are of great importance to modelers of dynamic systems for controls purposes. Electrical signals from the human sensory neurons received from electro-encephalograms (EEGs) and electro-cardiograms (ECGs) are encoded as time series. There are other important day-to-day activities such as the market outlook (bears or bulls), weather and climate changes, genetic sequences, etc., that are represented in this data form. Due to the need for extracting salient information from the trends, some of the recent-past methods that are applicable to these category are principal component analysis (PCA), independent component analysis (ICA), Neural Networks (NN) and filtering techniques such as Bayesian filters, Kalman filters and particle filters (Rao et al. (2009b)). Kalman filtering, wavelet packet transforms, and least squares support vector machines for instance, are used to predict wind power performance (Zuluaga et al. (2015); Wang et al. (2015)), while an analog ensemble method is applied to forecast solar power (Alessandrini et al. (2015)). Liu et al. (2015) predicts remaining state of charge of electric vehicle batteries based on predictive control theory. Hybrid genetic algorithms and Monte Carlo simulation approaches are applied to predict energy generation and consumption in net-zero energy buildings (Garshasbi et al. (2016)). For modern energy systems, a large number of subsystems is usually involved, for example, hundreds of wind turbines are closely collocated in a wind farm where the wind resource is similar and the conditions of each are analogous in terms of the power transmission

to the power system. As a result, there is a relationship among the wind turbine outputs, and the characteristics of their spatial interactions can be potentially applied for prediction (Jiang and Sarkar (2015)) and design optimization. The prediction approaches discussed above can be viewed as methods of exploring temporal relationships. Spatial and temporal relationship widely exists in energy systems (Jain et al. (2014); Liu et al. (2010); Jung and Broadwater (2014); Kwon (2010)) These have to a large extent formed the basis for dimensionality reduction for time series analysis. There are also a host of methods that deals with the frequency domain transformation, Fourier and discrete Fourier transforms of time series that have proven useful for analyzing stationary signals such as determining prominent modes of data. In this regard, we have developed a novel algorithm called hierarchical symbolic dynamic filtering (HSDF) (Akintayo and Sarkar (2015)) based on recently developed symbolic dynamic filtering (SDF) (Ray (2004); Sarkar et al. (2013a)) to learn models from streaming type data in an unsupervised manner (i.e., without knowing how many unique characteristics or classes are present in data). The algorithm is an observable Markov process that was shown to outperform the switched linear dynamical systems (SLDS) (Fox et al. (2008)) for unsupervised feature classification. It takes advantage of the nonlinearity inherent in the SDF framework to discover features that can, in this case can be considered to be arising from switched nonlinear dynamical system. Also, some of the pioneers of SDF developed a Multi-scale Symbolic Time Series Analysis (MSTSA) for characterizing seismic activities monitored by unattended ground (Sarkar et al. (2015a)).

2.4.3 Spatiotemporal data

While the spatial and temporal cases have been considered separately, the research community finds the need for advancing the architectures to merge both considerations for a more versatile physical application. The category can also be seen as moving points (Erwig et al. (1999)). Some of the applications of such framework includes, but are not limited to, complex systems such as nuclear power plants (Jin et al. (2011)), coal-gasification systems (Chakraborty et al. (2008a)), ship-board auxiliary systems (Sarkar et al. (2013c)) and gas turbine engines (Sarkar et al. (2008);

Gupta et al. (2008)). Object tracking, global positioning system, autonomous guidance and navigation systems also fall under these richest classes of data. Their analyses utilize the benefits of high dimensionality in space of images and maps, while varying in another dimension (temporal) leading to a multi-dimensional perspective. The associated storage and management require is equally more challenging than the previous categories. This current research's novelty would be in part, the developed tools' application to complex spatiotemporal engineering and plant science datasets such as those already considered by Sarkar (2015). It also aims to improve the learning schemes of the spatial as well as temporal components for extracting richer features of features (Salakhutdinov and Hinton (2012); Srivasta and Salakhutdinov (2012)). While the deep network methods such as convolutional networks and autoencoders have mainly been suited for hierarchical over-complete jointly convolved kernels learning (or parameters learning for lower dimensional space projections of the input data) from rich spatial data, their success in spatiotemporal data applications have been limited for applications where the frames in the long-range dependent sequences are not sampled from distributions that are independent and identical. Examples of such spatiotemporal data in traditional computer science applications are semantic labeling, language translation that have potential applications in autonomous guidance vehicles and self-driving cars. Hidden Markov models, HMMs (Rabiner (1989)) that were originally suitable for modeling sequences are limited by model complexity of dynamic hidden units, these are giving way for superior learning ideas such as the recurrent neural networks, RNN (Lipton et al. (2015)) and its stacked variant, long short-term memory, LSTM (Gere et al. (2000)) that had been reported (Siegelmann and Sontag (1991)) to theoretically satisfy the Turing completeness property. Interestingly, recurrent neural networks also apply a similar notion to our proposed selectivity for information across sequence of steps in its one frame at a time analysis, rather than our spatial selectivity. The exploration of such spatiotemporal features has been shown to be efficient in wind speed forecasting problems (Tascikaraoglu et al. (2016); Jung and Broadwater (2014)). For spatio-temporal pattern recognition, an extension of HMM to Factorial Hidden Markov Model (FHMM) (Ghahramani and Jordan (1997)) parallelizes multiple Markov models in a distributed manner, and performs some task-related inference to ar-

rive at predicted observations. Deep learning techniques have the capabilities for capturing the correlation at both multiple scales as well as dimensions (as in the known statistics). Feature extraction by machine and deep learning techniques are done directly from data, unlike the traditional extraction techniques, such as scale invariant feature transforms (SIFT) (Lowe (2004)), histogram of oriented gradient (HOG) (Dalal and Triggs (2005)) where the features have to be mechanically crafted into the machine, or their improvements described in symbolic dynamic filtering techniques (SDF) (Akintayo and Sarkar (2017)) for features classification. The most basic, but important first approach to addressing the problem employed classification of frames present in the video. A lot of work has been done in the area of classification using the randomized isolated frames with convolutional neural network (CNN) (Krizhevsky et al. (2012b); Simonyan and Zisserman (2014)). While such techniques are considered static-type classification, techniques for modeling dynamic classification in videos conditions are now relatively well understood (Wu et al. (2015)).

2.5 Development of Neural Networks

Up to this point, neural networks have been mostly highlighted as a potential framework for extracting features, data compression from spatial, temporal and spatiotemporal data. This section onwards are devoted to the underlying principles of neural network algorithms, and their extension to deep learning models. Artificial neural networks are biologically-inspired (Dalva et al. (1997)) techniques developed with the aim of preserving local neuron (also units) connections as well as being sensitive to changes in data orientations. Deep networks are a recent extension of artificial neural network algorithms that trains several layers to attempt learning good internal representation of features in purely layer-wise way (Hinton et al. (2006); Bengio et al. (2007); Ranzato et al. (2007); Vincent et al. (2008)). The manner in which this is done is similar to how human beings divide-and-conquer complex activities in a hierarchical fashion. Deep learning-based approaches are immensely attractive because of they competitive effectively and/or outperform many other state-of-the-art machine learning tools for a large variety of computer vision applications such as object recognition (Krizhevsky et al. (2012a)), scene understanding (Couprie et al. (2013)) and the

detection of occlusion edges (Sarkar et al. (2015d)). The authors of (Tenenbaum et al. (2011)) investigated how human cognitive processes leading to learning so much from little has link to the manner of probabilistic inference made at each level in a hierarchical fashion with layers of flexible and adaptive structures. (LeCun et al. (1998a)) reported Hubert and Wiesel work that inspired the discovery of how the characteristic benefit of the primary visual cortex of cats are the phenomenal basis for the best performing algorithms.

However, before focusing on specific deep learning algorithms, a categorization of machine learning algorithms are presented here. The plethora of machine learning methods with interwoven steps makes categorization quite complicated. In the same vein, a list of the methods are also dynamic such that an attempt to categorize them may leave out several other important ones. Two fundamental components of training a machine learning framework are the feed forward activation of the units of the current layer and either a joint back-propagation of the errors over all layers or a layer-wise pre-training that samples from the activations in a top-bottom generative fashion. While neural networks have several definitions based on authors view, the most mathematically concise definition of a neural architecture by Rojas (1996) is provided in the following.

Definition 2.5.1 (**Neural Network**)**.** It is a 4-tuple (I, N, O, E), such that I represents the input sites; N is the network of computing layers and nodes; O is the output sites; and E is the weighted (weights, {w}) and biased (biases, {b}) edges.

Definition 2.5.2 (**Neural Network Edges**)**.** The edges of the network are a 3-tuple, E = (u, v, θ), such that $u = I \cup N$; $v = N \cup O$; and the network parameters $\theta = (w, b) \in \mathcal{R}$

It is to the units of the computational graphs that the nonlinear activation functions are applied. These functions are usually selected to satisfy some general differentiability criterion, while the real-value network parameters are randomly initialized. Consider the simple perceptron (Rosenblatt (1958)) modeled mathematically in Equation 2.1 in Figure 2.2, the training rule – sum and thresholding – associates the output of a 2-layer network to the inputs.

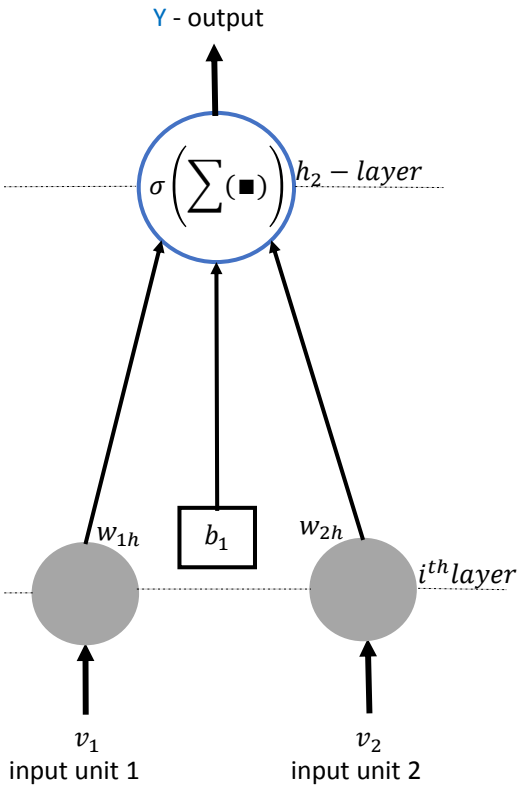


Figure 2.2: A simple perceptron training rule schematics.

$$y = \sigma\left(\sum_l (w_i v_i) + b\right) = \begin{cases} 1, & \text{if } z > threshold \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Where v_i's are the input units to the network, and $z = \sum_l (w_i v_i) + b$, the weighted sum that is activated by a function, σ on the weighted, (weights, w_i) sum of the inputs added to a constant bias, b to give the output y. The power of the learning rule is bestowed by the activation that is in the case of a perceptron, a binary thresholding function, ($\sigma_{perceptron}$). The weights are therefore the relative loading of the network, while the function for a perceptron is a thresholding-based rule. By design, the perceptron is able to resolve most linearly separable problems CH1 (2016).

Definition 2.5.3 (**Linearly separable problems**)**.** These are problems for which a hyper-plane is sufficient to separate the activated space from the suppressed ones.

However, when the Boolean logics examples are considered, the 'OR' and 'AND' gates are linearly separable. The simple plane alone may not be easily generalizable to the 'XOR' gate where a function that is more complex than a simple hyper-plane is required to discriminate the units. An example of such complex association was previously tailored towards engineering the units at the input layer, with the inclusion of 'NOT' gate to the former in order to yield a 'XOR' gate model. One important notice is that the neurons in the perceptron rules are hard-fired (0 $\rightarrow$ 'off' and 1 $\rightarrow$ 'on') leading to instabilities in the parameter θ(W, b) learned due to jump discontinuities in the feedback network structure. This led to the development of artificial neurons whose activations, for example rectified linear unit (ReLU) in Figure 2.3(a) and sigmoidal function in Figure 2.3(b) are continuous almost everywhere and at least piecewise differentiable. Figure 2.3 also shows how the Gaussian skews in favor of different activation level along the activation profile for the units. We note that an advantage of ReLU is its ease in training compared to other nonlinearity types because the activations of each neuron is piece-wise linear, such that each of the real values of the function have equal chances of being activated beyond zero value as shown in the the Gaussian to its left in Figure 2.3. Other benefits of ReLU that are widely known are that the activation does not saturate at the top and ReLU helps a network to reduce the debilitating effects of vanishing or exploding gradients.

Given these adaptive neurons, the addition of several more input units, and therefore more parameters became one of the most viable ways to model a complex hyper-plane discriminability. This is severally referred to as increasing the width of the neural networks, and it serves as the basis for more complex discriminatory classifiers such as the support vector machines (Burges (1998)). However, these kinds of networks have the associated cost of exponentially increasing number of parameters required for training the network. The more parsimonious approach to modeling complex relationship was found to be in favor of increasing the depth of the network, since the parameters then link the previous layers to both develop more complex relationships from previous layers' simpler features while also reducing the number of parameters to limit over-fitting. There is no debate (Eldan and Shamir (2016); Segzedy et al. (2015)) on the benefits of increasing the

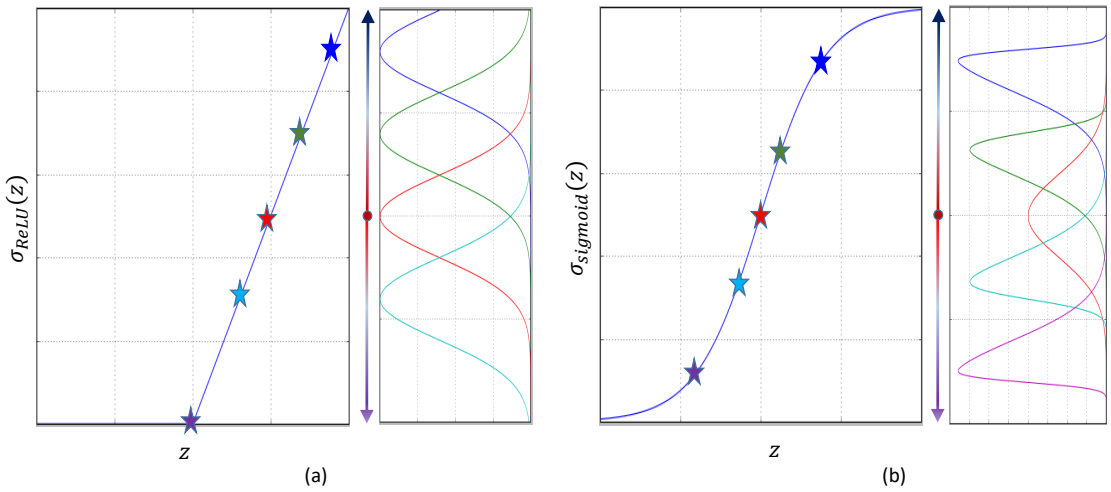


Figure 2.3: Artificial neural network activation examples: (a) rectified linear unit ($\sigma_{relu} = max(0, z)$) (b) sigmoidal activation ($\sigma_{sigmoid} = \frac{1}{1+e^{-z}}$) and their respective Gaussians depicting their relative activation strengths around the values.

width or depth of the network shown in Figure 2.4, in the face of many state-of-the-art performance recorded since the successful training of deep networks (Hinton et al. (2006); Bengio et al. (2007)). Perhaps, the most striking advantage is their ability to mimic the cognitive manner in which human beings learn to abstract complex problems and/or generalize when the data is sparse (Tenenbaum et al. (2011)). Also, it has been established that each layer of deeper networks is able to individually disentangle some underlying factors of variation that describe the complex space of the data (Bengio (2009)) in a hierarchical manner.

One of the most intuitive background to the difficulty of initially training deeper architectures was given in Chapter 5 of Nelson (2015). Our experience (Akintayo et al. (2016a,b)) on training deeper network for plant science and engineering applications lend credence to the hypothesis. Some of the problems that arises are: differences in learning rates of various layers, instability in gradients of lower layers requiring a balance between enhanced ability and delicateness in training. For emphasis, training a network has been described to be composed of two actions: the forward activation and the backward - optimization.

Thus far, only the algorithmic aspect of the network has been described. It is also pertinent to provide a brief introduction to the implementation capability required for this research. The

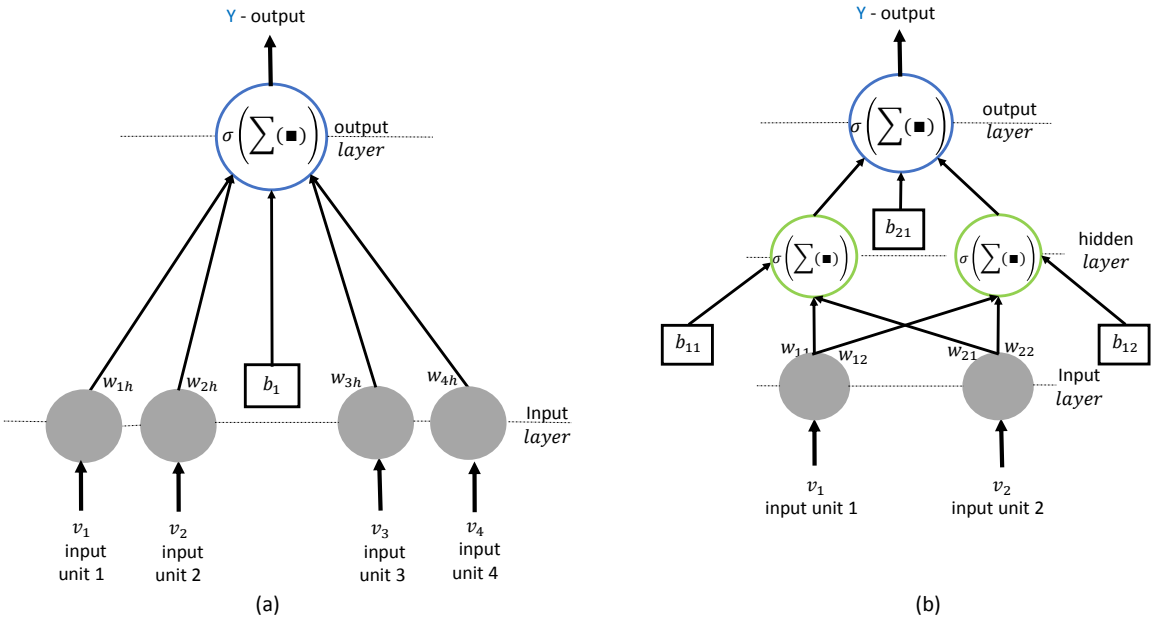


Figure 2.4: Schematics showing comparison between, (a). wider network and (b). deeper network.

hardware related issue in the forward network training of the large volume of task-related data to properly extract meaningful features has been favored (Bergstra et al. (2010b)) by parallel platforms by compute nodes of graphics processing cards. A benefit of which is the faster computation over sequential training procedure. Therefore, part of the training mechanism is required for loading the data from the memory of the host (central processor) to the nodes of the GPU. Although the nodes of GPUs are faster, they actually have lower memory capacity limiting their ability for whole data training. In any case, there is still an open question on what memory size would be required for deeper networks, since more and more available dataset implies better training. In this light, there are different training schemes, namely: the deterministic full data training, the single data stochastic training and the middle ground – mini-batch training. The advantage of the most common mini-batch training is to balance the size required by the graphics unit while ensuring the robustness in randomly chunked examples that are utilized for training the network each time.

Network learning ability is mainly influenced by the backward pass of the error which are utilized to fine-tune the parameters in the training feedback loop. Two major choices in the feedback loop

of the neural network are the loss function and the optimization scheme that evaluates the 'best' value of the function while the weights and biases are modified. We concentrate on these two important components, their consequence on training and other underlying factors that influence their choices in real-life problems.

2.5.1 Loss functions

The loss or objective function is a function that models the variables or parameters of, and constraints to a certain goal in mathematical terms. Some other applications of such functions are in decision making, game theory and operations research techniques. Adequate design of this function is the backbone of the results achieved in many of the applications. Improvements in the tasks' results have been reported (Zhao et al. (2016); Chen and Wang (2013)) when the objective functions are designed to match the performed tasks. In our case, we include the selectivity condition, that is formalized in Section 4.2, so that the tasks are converted to match the functions to be optimized. The following are the common types of loss functions that are implemented in machine learning algorithms. For all the loss types, let $y^o = F(v^o; \theta)$ be the network outputs and t^o, the known targets, the subscripts 'o' represents the total sample divided by number of mini-batches and F is a combination of layer-wise of activation functions that propagates the input units to the output units.

2.5.1.1 Squared error loss

Squared error loss is one of the most common loss type, commonly used in gradient descent algorithm. It is an L2-norm of the network output and the target that ensures convexity and analyticity properties making (Zhao et al. (2016)) it suitable as objective function, $J(\theta)$. It is widely used in regression models, such as facial marker detections (Jafri and Arabnia (2009)).

$$J(\theta) = \frac{1}{2O} \sum_{o=1}^{O} (t^o - y^o)^2 \tag{2.2}$$

The loss function is almost always minimized.

2.5.1.2 Cross-entropy

Cross-entropy function is a suitable function for training autoencoders (Vincent et al. (2008)) and unsupervised networks (Salakhutdinov and Hinton (2009)) as well as in many supervised networks that will be shown. The cross-entropy functional between two target distribution may be defined as the sum of the difference between the entropy of the target distribution and the Kullback-Liebler (Kullback and Liebler (1951)) distance of the distribution of the output from that of the target. As a loss function, it applies non-uniform penalties to the net output values relative to the target. The penalties are non-uniform because the targets that have, at the current step successfully mapped the output are scaled at a different rate from those that have not satisfied the mapping. The function is widely used for classification problems as it is able to minimize wrongly selected classes. A proof of its derivation when the network is activated with a logistic sigmoid function was given in (Chapter 3 of Nelson (2015)) by relating the gradient of the error with respect to the activation to its gradient with respect to output. The binary version of the functional is shown in equation 2.3

$$J(\theta) = \frac{1}{O}\sum_{o=1}^{O} -(t^o log(y^o)) - (1-t^o) log(1-y^o) \tag{2.3}$$

The categorical cross-entropy functional generalizes the binary version to relating multiple output and target values.

2.5.1.3 Hinge loss

Hinge loss is also popularly used as squared hinge loss. It is widely in support vector machines (Burges (1998)) for classifications. The loss functional was described by Chen and Wang (2013) as the maximum margin objective function since it was meant to separate classes of support vectors using a hyper-plane. It has a mathematical expression of the form:

$$J(\theta) = \frac{1}{2}W^T W + C\sum_{o=1}^{O} max(\big(1 - W^T y^o t^o), 0\big) \tag{2.4}$$

The classes $\in \{-1, 1\}$ and θ is mainly the included weights, W.

Yet another loss function in the category of support vectors classification is the ranking loss, (Chen and Wang (2013)). Also, a multi-scale version of the structural similarity index measure, MS-SSIM that we (Lore et al. (2017)) have previously used as performance metric, have been proposed (Zhao et al. (2016)). It is given by the weighted sum of SSIM and the absolute error, and yield a cost function that addresses the sensitivity of the human visual system to local structures in image processing tasks.

2.5.2 Optimization schemes

The procedure that implements finding the "best" parameters from the cost function while maximizing the generalizability of the network is the optimization scheme. There are a host of choices involved in deciding the appropriateness of schemes for particular objective function based on the function's properties already described in subsection 2.5.1.1. However, the resulting effects of these properties make some of the factors of the optimization function intractable or undefined. Yet, other factors like the computation time and complexity are also commonly evaluated for the schemes. Whether it is able to find the solution in a reasonable amount of time, complexity and around an acceptable small ($\sim$say ϵ) neighborhood of the true solution are important. Optimizing deeper networks definitely pose greater challenges (Goodfellow et al. (2016)). The objective functions for deeper networks are usually a surrogate to the true losses of the actual functions, with the hope that the functions may buy the advantage of more generalizability in test situations. Usually, the aim is to minimize the expectation of the objective function over the data generating distribution, P_{data} from which our observed finite samples are derived. Therefore,

$$\theta^* = \arg\min_{\theta} J(\mathcal{E}_{(v,y)\sim P_{data}}; \theta) \tag{2.5}$$

It is perhaps for this extra complication that the problem of going deeper with neural network had been elusive for long, before the success of the recent past decade. The true data distribution is intractable from the finite samples, but it is now approximated by an empirical distribution in the empirical risk minimization approach (LeCun et al. (1998b)). Optimization algorithms are broadly classified into Hessian-based or approximated and the Hessian-free types (Goodfellow et al. (2016)).

Analytic functions are usually amenable to the Hessian-based types while the Hessian-free (Martens (2010)) types are suitable for intractable functions. Here also, the choice between mini-batch and deterministic algorithms are to be considered. In deep networks, sampling based stochastic gradient descent optimization are favored (Goodfellow et al. (2015)) due to their better parameters update when they change in response to the error profile. Figure 2.5 shows the distinction between the different sampling available for the construction of the loss function.

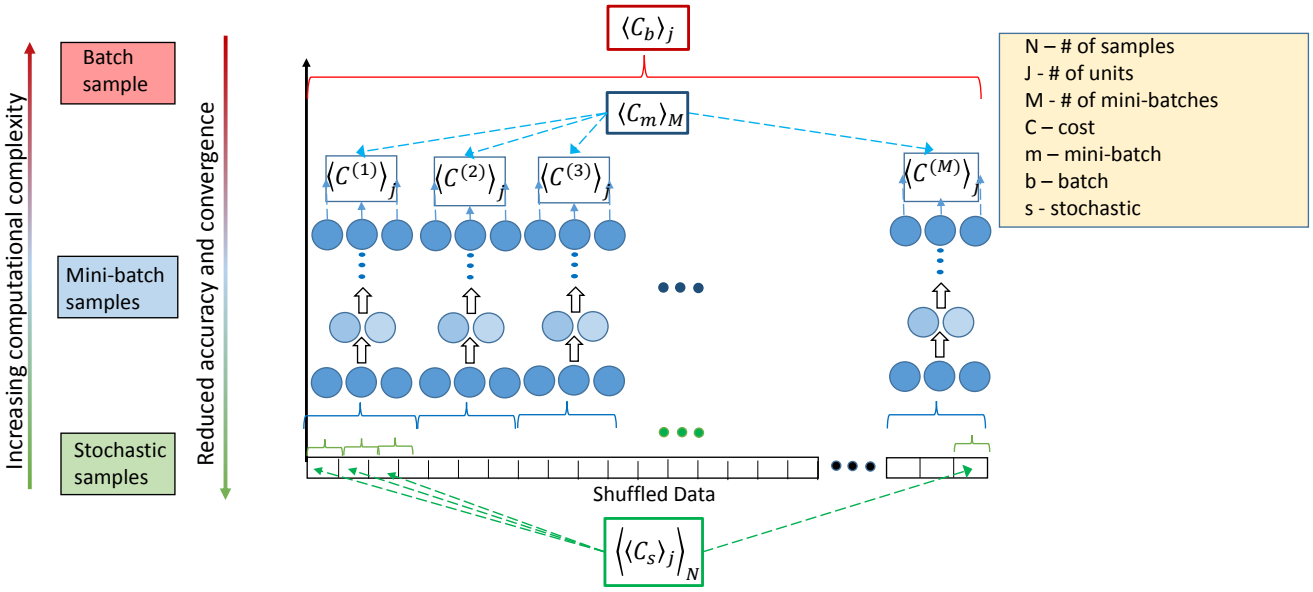


Figure 2.5: Data sampling techniques available for the stochastic gradient descent algorithm.

The authors, Goodfellow et al. (2015); Ruder (2016) reported that the stochastic gradient descent algorithms are faster to converge than the deterministic batch gradient descent because it eliminates the multiple update steps in the latter. Certain hyper-parameters that are explored further may play some important roles in balancing the learning stability, rate and convergence of the neural network training.

2.5.2.1 Learning rate

In gradient descent optimization algorithm, the choice of learning rate is important for efficient minimization of the loss function with respect to the parameters. The learning rate is the step

size in the gradient descent direction. If we assume that the error profile for some applications is convex (that is the optimum point is both a local and a global minima), Figure 2.6, then we have a well-behaved, smooth optimization problem for which extra consideration of the following factors are still required. Constant step sizes have the effects that a large step sizes lead to divergence or instability close to the basins or optimum points. However, a low rate of learning is usually inefficient to converge to the optimum points and invariably, it leads to poor performance called under-fitting (Martens (2010)). In order to balance the side effects introduced at both ends, varying, mostly reducing, step sizes have been introduced. The assumption behind this is that the profile of the error curve starts steeper and then would normally become gentle near the optimum points. Adaptive step sizes where k is the number of epochs, with properties, $\sum_{k=1}^{\infty} \epsilon_k = \infty$ and that $\sum_{k=1}^{\infty} \epsilon_k^2 < \infty$ (Nelson (2015)), are now included to enhance the step size profile by interpolating on the number of epochs to complete learning. The trainer who has some prior knowledge of the expected error distribution usually specifies the profile. The main bottleneck in this choice is the requirement of a preset adaptation, which usually has to be approximated for new training or application. SGD is sensitive to the choice of the learning rate, and it has been observed (Bottou and Bousquet (2007)) that however rapid the convergence, the upper bound (Wolpert and Macready (2012)) of error decrease is $\mathcal{O}(\frac{1}{k})$ and the scale of deep network optimization would make the batch GD convergence rate to result in over-fitting.

2.5.2.2 Momentum factor

Another hyper-parameter $\in$ [0,1] that factors in parameter changes trends is the momentum factor. It monitors the trends of exponentially decaying moving average of the past gradients in order to continue in that direction. Usually, the stability of the parameter vector update at the final stages is controlled by the momentum hyper-parameter. Mathematically, the hyper-parameter helps to condition the Hessian (or approximated Hessian) matrix; it averages out the variation in the trends of the gradient descent algorithm and stabilizes the training by when increased steadily to the end of optimization (Hinton et al. (2012)).The large-scale stochastic gradient descent algorithms

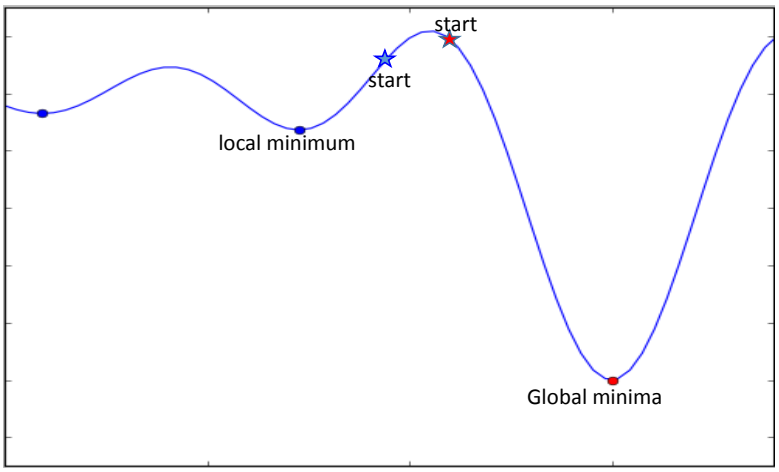


Figure 2.6: Local and global minima in dots, and their respective starting points are starred on an example function.

in neural networks for most applications are not convex (globally) as shown in Figure 2.6 because of model non-identifiability. Non-identifiability problem is worsened by the depth of the network; it is described by Goodfellow et al. (2016) to be due to the non-uniqueness of units in all the layers. If there are I layers, each having J units, then there are $J!^I$ symmetry in the weight space which leads to error functional to having several local minima. However, it is possible to leverage the trade-off between the complex, powerful and easily optimizable version where they have been decomposed to locally convex schemes (Chen and Wang (2013)) since the symmetry assures that the cost of all the several minima are similar and many of those represent the same global minimum.

A host of other attempts at optimizing such functions have concentrated on regions around a known convex hull (Candes et al. (2015); Wang and Giannakis (2016)). In high dimensions, (Martens (2010); Chen and Wang (2013)) propose that flat regions such as saddle points pose more threat than local minima. Logical and intuitive attempts at recovering from such points are: random initialization of the parameter value, addition of the patience hyper-parameter, among many other techniques.

2.5.2.3 Random Initialization

Random initialization can be imagined as a way of determining the global minimum from a finite set of local minimum that results from different random initializations (Glorot and Bengio (2010)) of weight parameters such that they are neither too large to prevent exploding gradients, nor saturating gradients for such activations as sigmoid (Goodfellow et al. (2016)). Lower values of weights on the other hand have reverse effect of vanishing gradient and as well as saturation while (Saxe et al. (2014)) proposed initializing with a scaled version of a random orthogonal matrix that enhances the applied layer-wise nonlinearities.

2.5.2.4 Patience factor

Patience on the other hand is a hyper-parameter that is included in the early stopping algorithm (LeCun et al. (1990)). It controls the number of epochs that the algorithm waits for an improvement before the current best local optimum in the direction of error minimization. The more important requirement is the need for breaking weight symmetry of such non-locality. It is highlighted to capture initializing weights that link (Goodfellow et al. (2016)) the units of the next layer to the current in the hierarchy, are randomly to be different. In generative approaches (Vincent et al. (2008); Salakhutdinov and Hinton (2009)) layer wise pre-training of the network ensure that the complex data space is transformed into a low-dimensional manifold with a set of pre-trained weights. In this process, (Bengio et al. (2013)) pointed out that simple few modes completely describe the underlying data are derived, and the learnt weights are suitable for initializing the network with better global optimality guarantee. These tricks having been developed independently, are now combined for use in deeper models. Therefore, the update scheme for standard optimization incorporating the tricks explained with m number of mini-batches as,

$$\partial\theta^k = \mu^k(\partial\theta^{k-1}) - (1-\mu^k)\epsilon^k \frac{1}{m}\sum_m \nabla_\theta[J_m(\theta)] \tag{2.6}$$

Then with that updated, the weight at the next epoch is updated via,

$$\theta^k = \theta^{k-1} + \partial\theta^k \tag{2.7}$$

Where the learning rates $\epsilon^k = f_l(\epsilon^0)$, f_l is the function adapting the step size given an initial learning rate ϵ^0, and ν is the momentum hyper-parameter given by,

$$\mu^k = \begin{cases} (1 - \frac{k}{K})\mu^0 + (\frac{k}{K})\mu^K, & \text{if } t < T \\ \mu^K, & \text{otherwise} \end{cases}$$

to reduce the effect of exploding gradients. $k = 1, \cdots, K$, K is the preset number of epochs, while μ^0 and μ^K are predetermined.

A modification provided (Sutskever et al. (2013)) proposed Nesterov gradient type momentum that was factored in before computing the gradient at every step to replace the update in the weight change in Equation 2.6 as follows,

$$\partial\theta^k = \mu^k(\partial\theta^{k-1}) - (1 - \mu^k)\epsilon^k \frac{1}{m} \sum_m \nabla_\theta [J_m(\theta + \mu^k \partial\theta^{k-1})] \tag{2.8}$$

It was shown to help the training to recover from the problematic regions of the error profile, as well as permitting random initialization of the parameters.

2.5.2.5 Regularization factor

When the number of parameters are too many, most deep network training overfit. Similarly, with fewer number of parameters than is expected to capture the variabilities, there is tendency to underfit. Deeper networks features are optimal when the right fit for the data is achieved. Figure 2.7 shows and example of a function and an attempt to determine the right fit given the knowledge of the function. Although, the data fit is desired, there are benefits in learning over-complete dictionaries (Elad and Aharon (2006)), bases functions (Lee et al. (2006)) or kernels (Krizhevsky et al. (2012a)) such that the goal of generalization by the networks are fulfilled. Among the backlashes of learning such over-complete dictionaries is over-fitting. Regularization is a classical method for reducing over fitting when the parameters required modeling the data feature scales higher than the true required number of parameters. While training, over-fitting is noticed at a certain epoch where the decaying validation error suddenly increases, and sustains the increase, beyond the training error that is also decaying.

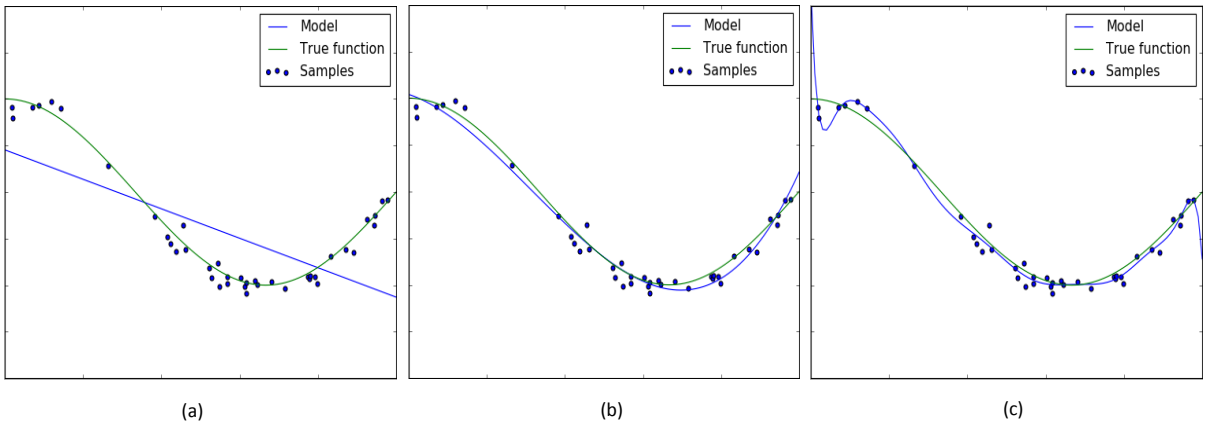


Figure 2.7: Schematics of different fits of an arbitrary function: $cos(1.5\pi x)$ plus random noise with, (a). underfitting – linear fit (b). approximate fit – 2^{nd} degree polynomial fit, and (c). over-fitting – 15^{th} degree polynomial fit.

One effect of over-fitting is that the weight array grows indefinitely for some units while many other units do not get filled. There is usually skewness in the parameter array that leads to computation problems, for instance, when taking the inverse of ill-conditioned arrays in the gradient-descent based optimization. Mathematically, this occurs when the quadratic term of the Taylor series expansion in Equation 2.9 exceeds the gradient term due to the Hessian matrix, H being out of bound especially when the expansion is in a perturbed neighborhood, $\delta\theta$ that is large relative to the parameters.

$$J(\theta + \delta\theta) = J(\theta) + \nabla_\theta J(\theta)^T \delta\theta + \frac{1}{2}\delta\theta^T H \delta\theta + H.O.T. \tag{2.9}$$

Where H.O.T. represents higher order terms. L2 and L1-norms are among the first known regularization methods that try to penalize terms the effect (Bishop (2006); Nowlan and Hinton (1992)). The actions of λ are to: decay the weights for reducing array-scaling problems and induce sparsity as well as robustness to irrelevant features:

$$J(\theta + \delta\theta) = J(\theta) + \nabla_\theta J(\theta)^T \delta\theta + \frac{1}{2}\delta\theta^T (H + \lambda I)\delta\theta + H.O.T. \tag{2.10}$$

In auto encoders (Vincent et al. (2008)) for instance, a low dimension manifold is desired from a high dimension data, limiting the divergence of the cross-entropy loss function from a specified hyper parameter value induces sparsity.

2.5.2.6 Dropout

Recent approaches have focused on reducing complex co-adaptation of neurons that learn similar features. This again leads to bad scaling of the parameter arrays because there are tendencies of the arrays being singular due to dependencies between the column of the arrays, and array ranks that are not full. There exist mathematical algorithms (Srebro and Shraibman (2005)) that deal with the algebraic analysis and manipulation of rank-deficient arrays with the inclusion of bounds of validity. In practice, dropout (Srivasta et al. (2014)) is known to reduce this complex feature co-adaptation by randomly setting some activation units to zero at each training epoch. The authors (Hinton et al. (2012)) described dropout as capable of providing efficient joint parameter learning in a multi-architectural framework that may be equivalent to averaging over the conditionals when several runs (Chen and Wang (2013)) of the algorithms are performed. They also have the advantage of re-normalizing only sets of weights that violate a constraint. For instance, max-norm regularization entails projecting the $||W||_2$ of every layer to a circle of selected radius, R when the $||W||_2$ exceeded R (Srebro and Shraibman (2005)). Improved results was obtained using dropout in the deep learning approaches over the then state-of-the-arts performances in handwriting, natural images, speech and news recognition (Hinton et al. (2012)).

Going forward, it is clear that the major contribution in training a deep network lies in the formulation of the objective loss as well as designing enhanced optimization scheme. It is noted that there exist several factors to be considered for the design to be considered and improvement. A few of the optimization algorithms that have been successfully applied to deep network are presented by Schaul et al. (2014); Goodfellow et al. (2016); Ruder (2016) that require further studies, analysis and testing are AdaGrad (Duchi et al. (2011)), RMSProp (Dauphin et al. (2015)), Adam (Kingma and Ba (2015)), AdaDelta (Zeiler (2012)). It is also important to keep track

of properties of the traditional optimization schemes such as the Levenburg-Marquardt scheme, conjugate gradient schemes such as, Fletcher-Reeves, Polak-Ribiere, Broyden-Fletcher-Goldfarb-Shannon (BFGS) and its limited memory version, coordinate descent method, polyak averaging and the recently introduced batch normalization (Ioffe and Segzedy (2015)) which is an algorithm that enhances re-parameterization to reduce covariate shifts whose effects are to saturate the activations at the lower layers, and thus require learning at a slower rate due to vanishing back-propagated gradient from the topmost layer.

2.5.3 Backpropagation algorithm

Backpropagation is a fast method for computing the gradient and examining how the parameter changes in preceding layers affect the output. With the increasing benefits of training deeper network, and associated topological complexities in parameter training, backprop algorithm is a popular technique devised for understanding a network's parameters flow. It is an important learning module that was reformulated by Rumelhart and McClelland (1986). It aids in learning the parameters by accounting for how changes in each parameter affects the output of a multi-layered neural network (LeCun et al. (1998b)). Although connections in the brain appear to be unidirectional such that all the units are locally related to the weight, unlike what is obtainable in back propagation. Backpropagation starts with the gradient descent algorithm that minimizes the cost function(e.g. average of the sum of squares error) relating the target and the network output. The original implementation then decomposes the gradient of the cost function with respect to the parameters (weights and biases) linking individual units in a layer via the chain rule to become the product of: the differential of the cost function with respect to the activation output; the differential of the activated output with respect to the input; and the differential of the input from previous layer with respect to the parameters. It is basically trying to determine the combinations of weight that leads to minimum error magnitude given by some functional of the target and last layer activation. Backpropagation is sometimes described with four equations (Nelson (2015)) to also include how the error in the layer above affects that in the layer below through the linking weights

and the weighted sums. The benefit of back propagating in the weight space can be understood to speed up changes to final state because it is a normal direction to a sigmoid activation space in Chapter 7 of Rojas (1996).

2.6 Learning Architectures

Having developed a generic neural (deep) network and the various considerations, this section is intended to focus on the details of the architectures of neural networks that have been useful to our cyber-physical problems. The more popular deep learning architectures are the denoising and convolutional autoencoders, convolutional networks (CNN), the energy-based (restricted) Boltzmann machines (R)BMs and the long-range dependency learners – RNNs. In the following section, the architecture that were mostly implemented will be discussed.

2.6.1 Stacked sparse denoising autoencoder (SSDA)

SSDAs are sparsity-inducing variant of deep autoencoders that ensures learning the invariant features embedded in the proper dimensional space of the dataset in an unsupervised manner. Early proponents Vincent et al. (2008) have shown that by stacking several denoising autoencoders (DA) in a greedy layer-wise manner for pre-training, the network is able to find a better parameter space during error back-propagation.

Let $\mathbf{y} \in \mathcal{R}^N$ be the clean, uncorrupted data and $\mathbf{x} \in \mathcal{R}^N$ be the corrupted, noisy version of $\mathbf{y}$ such that $\mathbf{x} = \mathbf{M}\mathbf{y}$, where $\mathbf{M} \in \mathcal{R}^{N \times N}$ is the high-dimensional, non-analytic matrix assumed to have corrupted the clean data. With DA, feed-forward learning functions are defined to characterize each element of $\mathbf{M}$ as follows:

$$h(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\hat{\mathbf{y}}(\mathbf{x}) = \sigma'(\mathbf{W}'\mathbf{h} + \mathbf{b}')$$

where σ and σ' denote the encoding and decoding functions (either of which is usually the sigmoid function $\sigma(\mathbf{s})$ or $\sigma'(\mathbf{s}) = (1+\exp(-\mathbf{s}))^{-1}$ of a single DA layer with K units, respectively. $\mathbf{W} \in \mathcal{R}^{K \times N}$ and $\mathbf{b} \in \mathcal{R}^K$ are the weights and biases of each layers of encoder whereas $\mathbf{W}' \in \mathcal{R}^{N \times K}$ and $\mathbf{b} \in \mathcal{R}^K$

are the weights and biases for each layer of the decoder. $h(\mathbf{x}) \in \mathcal{R}^K$ is the activation of the hidden layer and $\hat{\mathbf{y}}(\mathbf{x}) \in \mathcal{R}^N$ is the reconstruction of the input (*i.e.,* the output of the DA).

2.6.2 Convolutional networks

Convolutional networks (cnns) (Jarrett et al. (2009)) are discriminative models that rely primarily on local neighborhood matching for data dimension reduction using nonlinear mapping (i.e. sigmoid, softmax, hyperbolic tangent). The discriminative advantage of convolutional networks (Krizhevsky et al. (2012a)) are similar to that provided by maximum entropy markov models (MEMM) (Erdogan (2010)) added to a global relationship among observations as in the conditional random field (CRF) models (Domke (2013)). Each unit of the feature maps has common shared weights resulting in an efficient training having relatively – compared to fully connected layers – less number of trainable parameters (Krizhevsky et al. (2012c)). Feature extraction and classifier learning are the two main functions of these networks (LeCun et al. (1998a)). However, in order to learn the most expressive features, we have to determine the invariance rich codes embedded in the raw data and then follow with a fully connected layer to reduce further the dimensionality of the data and map the most important codes to a low dimension of the examples. Many image processing and complex simulations depend on the invariance property of the convolution neural network stated by LeCun and Bengio (1995) provided by the pooling layers. This aids to prevent in part, the problem of over-fitting models to training samples as expressive codes are learned from data. The feature maps are able to preserve local neighborhood patterns for each receptive field as with over-completeness dictionary (Aharon et al. (2006)). A full and detailed review was provided by LeCun et al. (1998a), where the authors noted the advantage of local correlation enforcing convolution before spatiotemporal recognition. For efficient learning purposes, convolutional networks are able to explore the benefits of distributed map-reduce frameworks (Fung and Mann (2004)) to leverage large training data as well as multi-GPU computing. With these benefits, the winners of the ILSVRC 2012 (Krizhevsky et al. (2012a)) utilized a large network of 8 layers with 2 GPUs while training with the same architecture provided by LeCun et al. (1998a) to achieve the then

best position. Subsequently, GoogLeNet (Szegedy et al. (2015)) and other authors (Simonyan and Zisserman (2015)) have also reported better performance with larger models found to be more related to the depth of the network. The layers that make up the convolutional network are the convolutional and the subsampling layers.

2.6.2.1 Convolutional layer

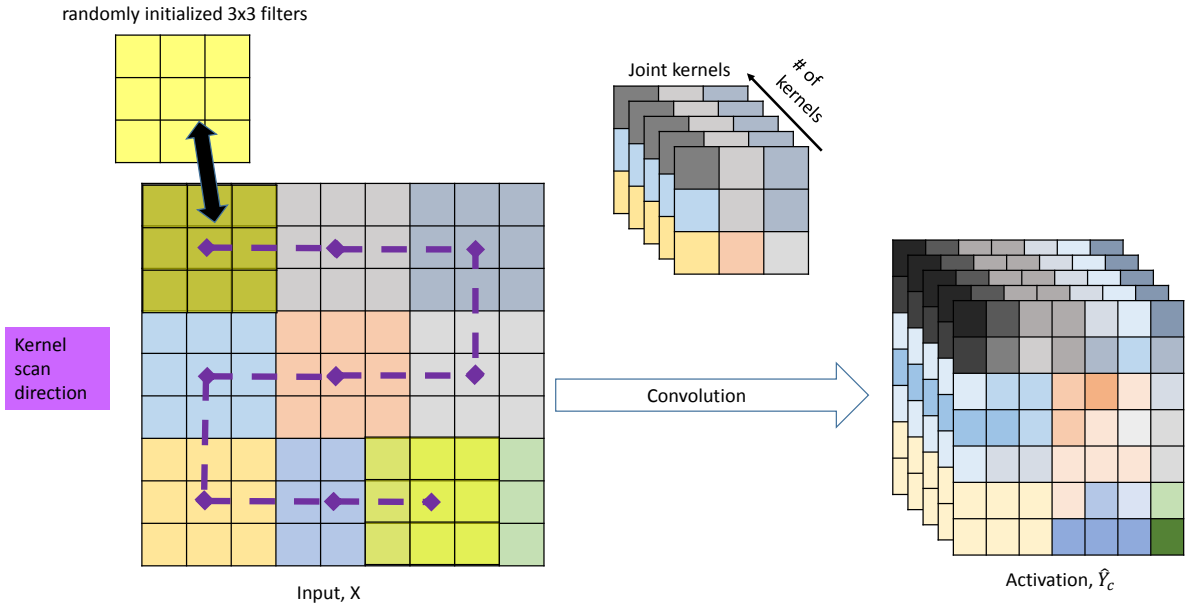


Figure 2.8: Schematics of the convolutional layers.

Given the input and output pairs introduced in Section 4.2, at each convolution layer (Figure 2.8) a chosen $(c \times c)$-dimension filter size is convolved with the patches to learn a z_o-dimensional feature map from which joint weight over the z_i-dimensional feature maps that are useful for enforcing local correlation is learnt to characterize all maps as follows,

$$\hat{Y}_{z_o(m-c+1)(n-c+1)} = C[X_{z_i mn} \star W_{z_i cc} + b_c] \tag{2.11}$$

where C is the squashing function, rectified linear unit used and $\star$ is a convolution operator of the joint weights, $W_{z_i cc}$, b_c the biases and input from previous layer, $X_{z_i mn}$.

2.6.2.2 Pooling layer

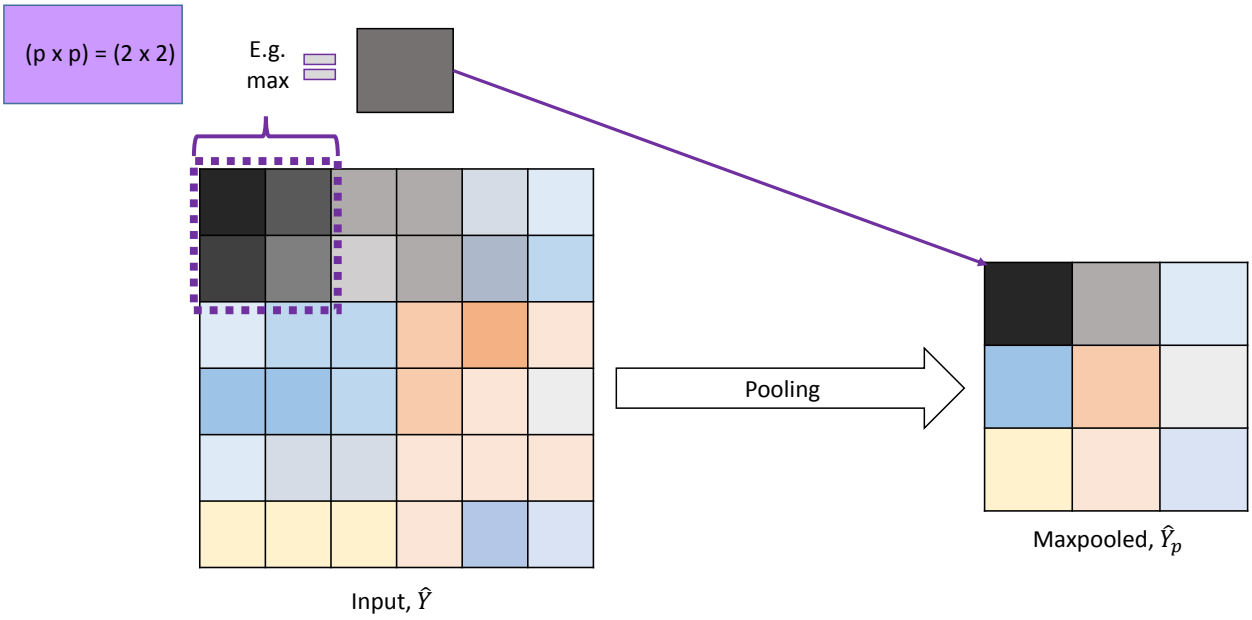


Figure 2.9: Schematics of the pooling layer.

This is a subsampling layer (Figure 2.9) that smartly reduces the data dimensionality in order to enhance computational efficiency. However, the major benefit of pooling is its introduction of feature invariance property. The layer is usually included after convolution layers to identify and propagate forward only the representative features in a local neighborhood. It ensures that the neurons activation in a locality favors low entropy activity of representing the assumed locally stationary distribution with the modal value. Physically, the most prominent of features (encoded from high dimensional data in convolution) in the local neighborhood are chosen to represent their feature-maps. In our formulations, maxpooling (Scherer et al. (2010)) had been chosen to propagate the maximally-activated unit as the representative for each of our chosen $(p \times p)$ neighborhood. Another well explored subsampling method is the mean pooling which we discovered to be influenced by the not well activated units in the neighboring feature maps.

2.6.2.3 Fully-connected layers

The topmost layers of a convolutional network is usually the multi-layer perceptron (Figure 2.10), especially when used in its traditional classification task. Its objective in image processing and object recognition are usually the end-to-end type low-dimensional encoding of image space. After the information compression performed on the low dimensional image manifold, a reconstruction of the original input space manifold is ensured. It has the disadvantage of requiring more parameters than the convolutional layers because of its fully connected structure, but it usually results in the improved detection results.

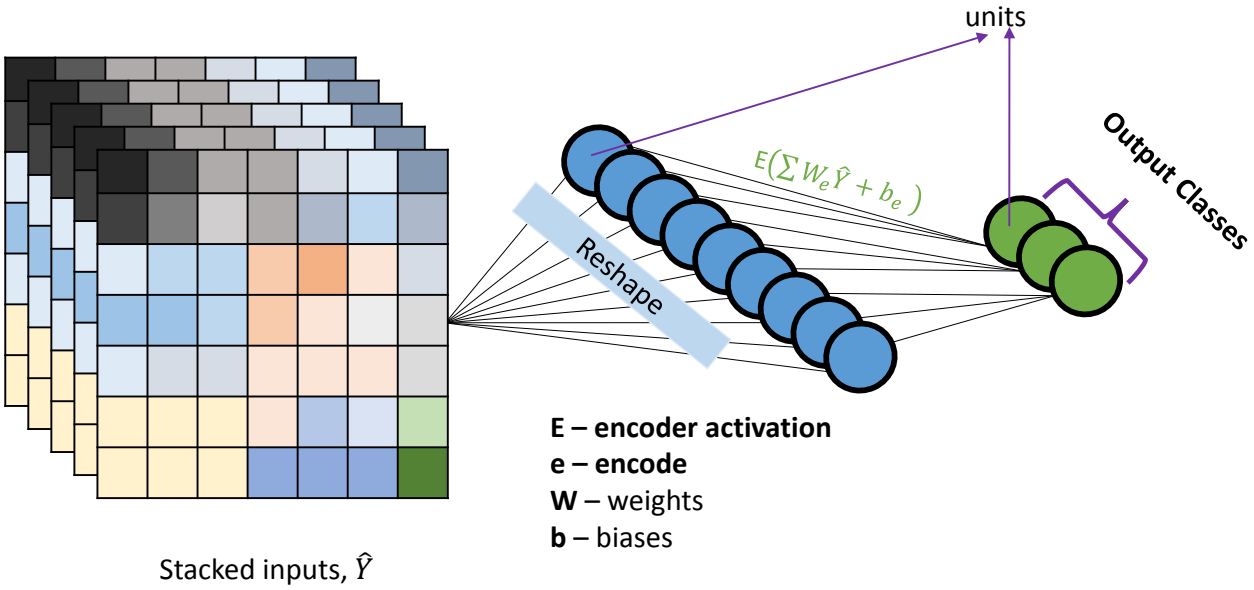


Figure 2.10: Encode layer for cnn classification

2.6.3 Autoencoders

Autoencoders are models that extend discriminative cnns to generative type models. Autoencoders are designed to take advantage of the few modal behavior provided by the higher level abstractions of deep networks. The high dimensional data is usual encoded with few representative

modes at the coding layer such that the decoded layer(s) are truly representative of the desired data features at lower dimensions.

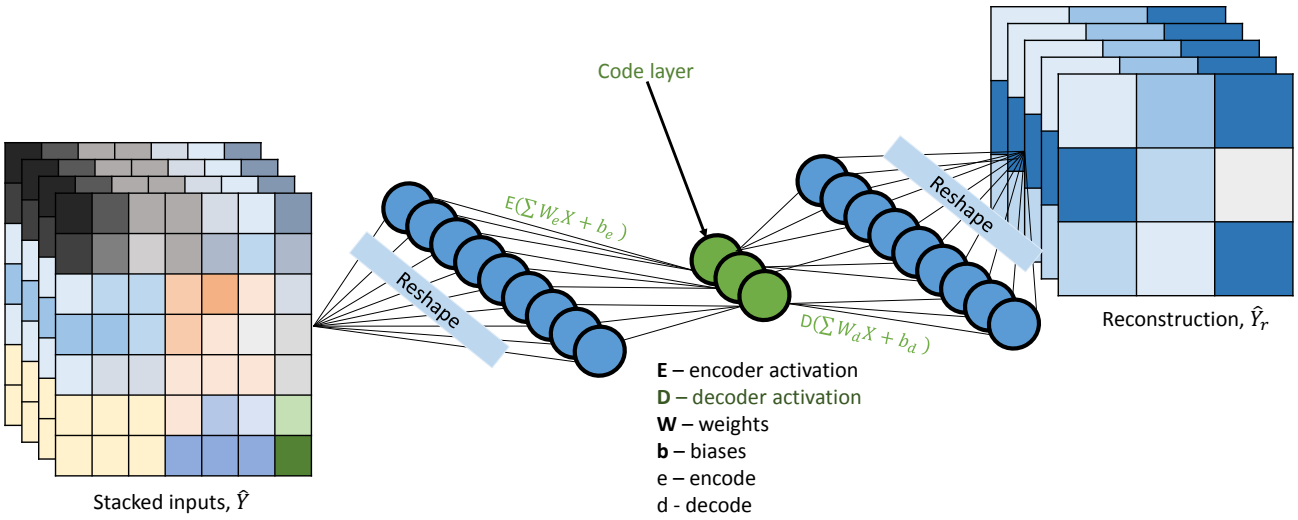


Figure 2.11: Pictorial description of the fully connected layers that make up an autoencoder network

The feature maps activation provided by the autoencoders are a part of the network which we termed the fully connected layers. In these layers, the feature maps from the previous convolution and subsampling layers are first flattened. Then, in order to reduce the number of parameters for the fully connected layers (Figure 2.11), combat the problem of over-fitting and to avoid getting trapped in local optima, some features are randomly left out with a dropout layers (Hinton et al. (2012)). Dropout in the hidden layer produces better results as it eliminates the necessity for regularization parameters used previously (Akintayo et al. (2016b)). A layer encodes the most important feature from the input layer (or previous layers if embedded in other autoencoders);

$$C = E(\sum[W_e\hat{Y} + b_e]) \tag{2.12}$$

and another layer reconstruct the useful features with

$$\hat{Y}_r = D(\sum[W_d C + b_d]) \tag{2.13}$$

Where E and D stands for the encoder and decoder activations, which are mostly sigmoidal, to admit the kind of objective training to be applied. b denotes the biases and W denotes the weights of the layer. The subscripts e and d indicates the encoder and decoder. $\hat{Y}$ is the input from previous layers. In training an autoencoder, the goal is usually to minimize the reconstruction error between the $\hat{Y}_r$ and the input, X by formulating a loss function (Subsection 2.5.1) that is most appropriate for the application. Note that using a sigmoidal activation for instance admits the cross entropy minimization function while some other functions work well with the squared-error loss function.

$$\theta^* = \arg\min_{\theta} J(\hat{Y}_r, X) \tag{2.14}$$

In that process, the parameters are updated to further reduce the error. Such models are however known to simply learn the identity functions when the goals are optimized. There are several variations of the autoencoder that have been explored to take advantage of the huge benefit of the parsimonious codes provided by the autoencoder. In denoising autoencoder where the input to the network is a corrupted version, $\widetilde{X}$ (by adding salt-and-pepper noise (Vincent et al. (2008)) to X) of the original input, X replaces it in Equation 2.12. Denoising autoencoders are then trained by minimizing the cross-entropy function loss. Contractive autoencoders (Rifai et al. (2011)) on the other hand penalize the regular autoencoder and the denoising autoencoder with a Frobenius norm of the Jacobian matrix of the activation with respect to the input.

2.6.4 Convolutional autoencoders

In such applications as object detection and semantic segmentation where the network is expected to out images from images, we extended the classification type convolutional neural network with an end-to-end type convolutional autoencoder, sometimes termed deconvolution nets (Zeiler and Fergus (2014)) or fully convolutional networks (Long et al. (2015)). Convolutional networks

have previously been augmented by energy models (Ning et al. (2005)) and symbol dynamic filtering (Sarkar et al. (2015b)) with improvements in performances. However, these improvements come with increase in number of trainable parameters, thus the network requires more data to train. Some other interesting ideas (Farabet et al. (2013); Pinheiro and Collobert (2014)) at enhancing such pixel-based semantic segmentation have been attempted with various improvements and backlashes. The key added layers to complete a convolutional autoencoder from a convolutional neural network are the deconvolution and the upsampling layers. The deconvolution layer is basically a convolution layer with the dimension of the feature map arrays for the input and output in Equation 2.11 transposed. It should also be noted that the sigmoid activations of E and D in Equations 2.12 and 2.13 are here replaced with the ReLU activation that suits the training of the overall cost function.

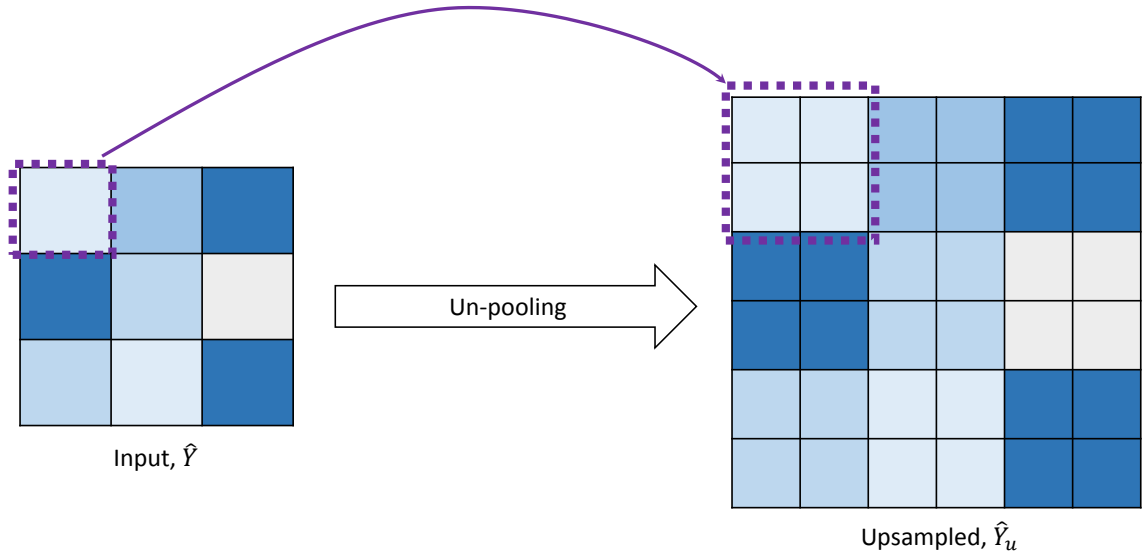


Figure 2.12: Schematics of the upsampling layer that distinguishes a convolutional autoencoder from a cnn.

In the upsampling layer, a reversal of pooling is done to restore the original image dimensionality by stretching and widening (Jones (2015)) the identified features from the filters of the previous layer. We consider it to be the reconstruction equivalence of the convolutional autoencoder network. It may also be imagined as an upscaling of the feature maps around the axes of symmetry where the reconstructed feature maps are optimized through the back-propagation algorithm.

2.7 Temporal Data Analytics

Some possible ways to analyze temporal data is the use of Markov chains and probabilistic graphical models. The success in analysis of many real world activities are facilitated by the elegant formulation of these techniques. In this context, we developed algorithms upon these concepts and have implemented the algorithms on observations from dynamical systems. Dynamical systems generate time series data. This series may either be continuous or discrete signals. These signals are shown to be too fine for most of the analysis and may contain unwanted disturbances that may affect the performance of a data-driven model. Techniques that are used to reject such unwanted artifacts from data are described then, the type of analyses techniques used in this research work are reviewed.

2.7.1 Discretization and symbolization

The signals generated by dynamical systems are either discrete of continuous in nature. Therefore, in the symbolic dynamic filtering (Ray (2004)), discretization of any continuous signals (or in some cases already discrete signals) into symbol sequences is a major first step in the SDF formulation. There are many ways of quantization (or partitioning) (Sarkar and Srivastav (2016); Sarkar et al. (2013b, 2012); Jin et al. (2009)) depending on the specific objective functions. Based on the characteristics of the measured signals, different symbolization techniques may be selected. The simplest approaches for partitioning are the uniform partitioning and maximum entropy, where data is partition based on either uniform count or uniform frequency methods respectively. These two methods were mainly applied to simple dynamical systems with data of less variance. The more recent discretization approaches include symbolic false nearest neighbor partitioning (SFNNP) (Kennel and Buhl (2003)), wavelet transform (Sarkar et al. (2013b)), and Hilbert transform-based analytic signal space partitioning (ASSP) (Subbu and Ray (2008)). Recently, a supervised partitioning scheme, i.e., maximally bijective discretization (MBD) (Sarkar et al. (2013b)) was proposed for modeling and analyzing complex dynamical systems. Unlike the other methods, MBD is able to maximally preserve the input-output relationship originating from the continuous domain af-

ter discretization in dynamical systems. After discretization of time-series data in the continuous domain, symbolization is conducted subsequently to establish the D-Markov machines.

The Figure 2.13 shows the symbol sequence generation in the form of PFSA using the two different methods, i.e., maximally bijective discretization and maximum entropy partitioning, respectively.

2.7.2 Symbolic dynamic filtering technique

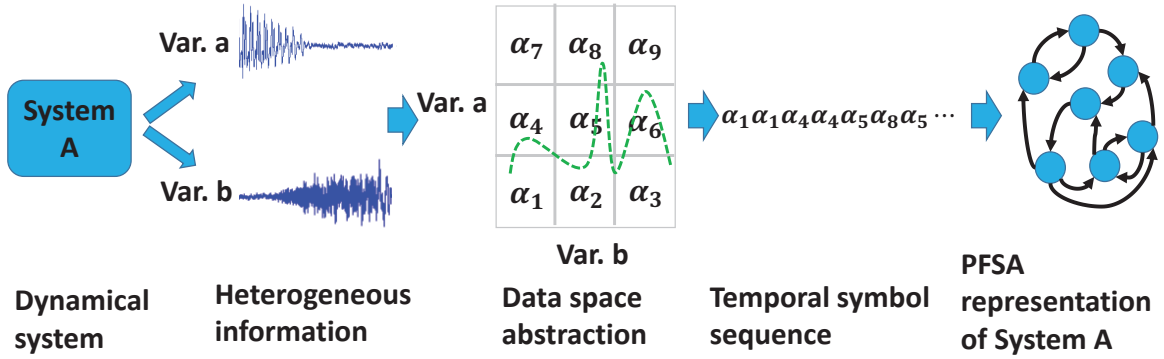
After the discretization process, PFSA models can then be generated from the symbol sequences (using PFSA models) that results from such quantization. SDF is built upon the relevant concepts of discrete dynamical systems in which discretization and symbolization are critical steps to convert observed continuous data into discrete symbol sequences. With SDF, dynamical systems can be studied in deterministic or probabilistic settings in terms of symbolic space by using language-theoretic approaches, e.g., shift-maps and sliding block codes. Given a suitably defined nonempty, finite set of symbols called alphabet Ξ, and nonempty, finite set of states Θ, the following definitions hold,

Definition 2.7.1. (Sarkar et al. (2014)) (DFSA) A deterministic finite state automaton (DFSA) is a 3-tuple $\mathcal{G} = (\Xi, \Theta, \delta)$ where,

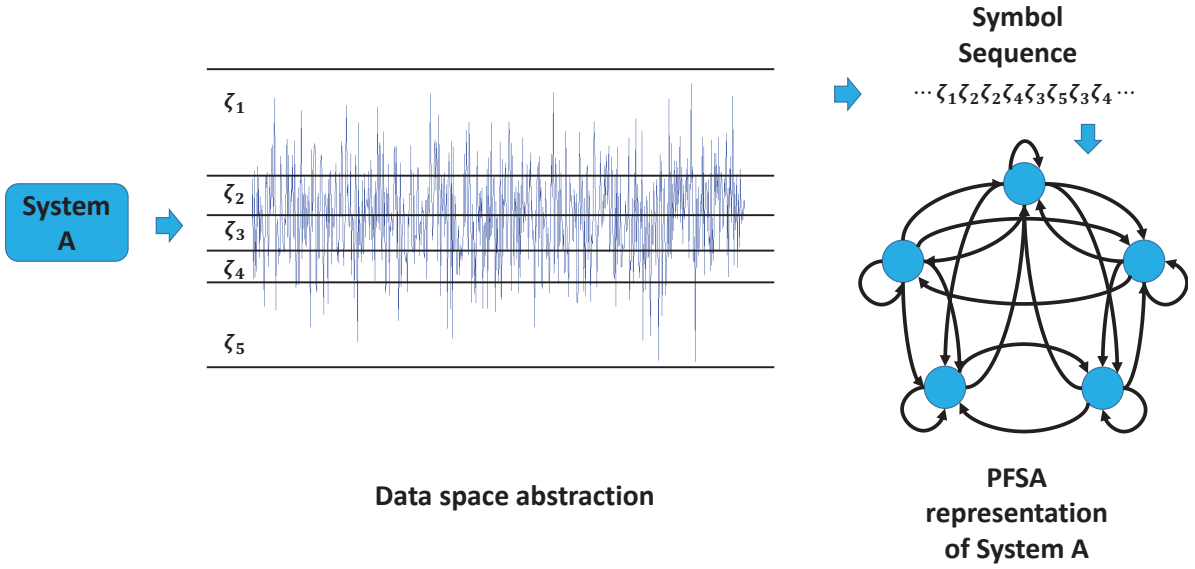
1. Ξ is a finite set of symbol alphabet and $\Xi \neq \varnothing (\textit{empty set})$;
2. Θ is a finite set of states and $\Theta \neq \varnothing$;
3. $\delta : \Theta \times \Xi \rightarrow \Theta$ is the mapping function for state transition;

while $\Xi^\star$ represents the collection of all finite symbol sequences from Ξ including the empty sequence ε.

Definition 2.7.2. (Sarkar et al. (2014)) (PFSA) A probabilistic finite state automaton (PFSA) is an extension to probabilistic setting from a DFSA $\mathcal{G} = (\Xi, \Theta, \delta)$ as a pair $\mathcal{K} = (\Theta, F)$, i.e., the PFSA K is a 4-tuple $\mathcal{K} = (\Xi, \Theta, \delta, F)$ where,



(a)



(b)

Figure 2.13: Illustration of generation of a PFSA using (a) maximal bijectively discretization and (b) maximum entropy partitioning for system A.

1. Ξ, Θ, and δ have the same definitions as in Definition 2.7.1;

2. $\Omega : \Theta \times \Xi \to [0, 1]$ is defined as a symbol generation function, i.e., probability morph function which are such that $\sum_{\xi \in \Xi} \Omega(\theta, \xi) = 1 \quad \forall \theta \in \Theta$, where p_{ij} indicates the probability of the symbol $\xi_j \in \Xi$ occurring with the state $\theta_i \in \Theta$.

δ is the state transition function and Ω is the state transition matrix. Nonlinearities in the time series are represented by a specific type of PFSA called the D-Markov machines (Mukherjee and Ray (2014)) where past depth D of symbols are considered for modeling the states as given by $|\Theta| \leq |\Xi|^D$ (Ray (2004)).The mapping $\delta : \Theta \times \Xi \to \Theta$ denotes a function that maps the transitions from a current state to a future state (or self transition) given the alphabet. A morph function $\pi : \Theta \times \Xi \to [0, 1]$ that satisfies the condition $\sum_{\xi \in \Xi} \pi(\theta, \xi) = 1$ is also considered. Based on the morph function, the non-negative ($|\Theta| \times |\Xi|$) state transition matrix Ω is defined as: $\Omega_{ij} \triangleq \pi(\theta_i, \xi_j), \forall \theta_i \in \Theta$ and $\forall \xi_j \in \Xi$. Online learning of an SDF model involves identifying this matrix. Note that initial state $\theta_0 \in \Theta$ of the quasi-stationary data represented by the $\mathcal{P} \triangleq (\Theta, \Xi, \delta, \Omega)$ has no influence on the state transition. However, a simple frequency count of the occurrence of symbols in the training string sequence at depth D, followed by normalization is used to derive the low dimensional encoding matrix Ω. Therefore, testing symbol strings that also follow the same quantization can be evaluated for similarity or difference with a PFSA represented by Ω.

2.7.3 xD-Markov machines extension of SDF

In order to establish SDF model for multiple time series, each time series due to the temporal features is associated with a symbol sequence (having different alphabet).

For SDF, a critical assumption is that we can approximate any symbol sequence generated by time series data as a Markov chain of order D (which is a positive integer). Such a Markov chain is called D-Markov machine.

Definition 2.7.3. (Sarkar et al. (2014)) (D-Markov) A D-Markov machine is an extension of a PFSA where the previous D symbols form a state as defined by:

1. D signifies the depth of a Markov machine;

2. Θ is a set of finite size for states with $|\Theta| \leq |\Xi|^D$, i.e., each state in a Markov machine is identified by some equivalence class of symbol strings whose length are D with symbols in Ξ;

3. $\delta : \Theta \times \Xi \to \Theta$ signifies the state transition function such that if $|\Theta| = |\Xi|^D$, then there exist any two symbols $\alpha, \beta \in H$ and $\gamma \in H^\star$ such that $\phi(\alpha\gamma, \beta) = \gamma\beta$ and $\alpha\gamma, \gamma\beta \in \mathcal{Q}$.

Remark 2.7.1. Based on the Definition 2.7.3, it can be concluded that a D-Markov machine is a stationary stochastic process $X = \cdots x_{-1}x_0x_1\cdots$, in which the probability of occurrence of a new symbol x_n is determined by the last D symbols, i.e., $P[x_n|x_{n-1}\cdots x_{n-D}\cdots x_0] = P[x_n|x_{n-1}\cdots x_{n-D}]$.

Given Ω, the state transition matrix, each entry of the matrix demonstrates the transition probability from one symbolic state to another. We give a simple example to illustrate this. Let the k^{th} state of one dynamical system A be θ_k^A such that the ij^{th} entry, i.e., π_{ij}^A of the matrix Ω^A indicates the probability of θ_{k+1}^A as i given that the previous state θ_k^A was j, i.e.,

$$\pi_{ij}^A := P\left(\xi_{k+1}^A = i \mid \xi_k^A = j\right) \forall k \tag{2.15}$$

Moreover, one can model an individual dynamical system making use of D-Markov machines. Because a D-Markov machine cannot capture the interaction dependencies for multiple systems or sub-systems in a large complex system, it has recently been extended to a xD-Markov machine, which was originally developed in order to obtain the internally causal dependencies among different systems or sub-systems. Different from correlation-based analysis, such a model can efficiently build-up and fairly generalize the causal dependencies (Chattopadhyay (2014)). The following gives a formal definition of xD-Markov machine.

Definition 2.7.4. (Sarkar et al. (2014)) (xD-Markov) Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the PFSAs which correspond to symbol streams $\{\mathbf{x}_1\}$ and $\{\mathbf{x}_2\}$ respectively. Therefore a xD-Markov machine is defined as a 5-tuple $\mathcal{R}_{1\to2} := (\Theta_1, \Xi_1, \Xi_2, \delta_1, \Omega_{12})$ such that:

1. $\Xi_1 = \{\Xi_0, ..., \Xi_{|\Xi_1|-1}\}$ represents the alphabet set of symbol sequence $\{\mathbf{x}_1\}$

2. $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{|\Theta_1|}\}$ is the state set which corresponds to symbol sequence $\{\mathbf{x}_1\}$

3. $\Xi_2 = \{\Xi_0, ..., \Xi_{|\Xi_2|-1}\}$ represents the alphabet set of symbol sequence $\{\mathbf{x}_2\}$

4. $\delta_1 : \Theta_1 \times \Xi_1 \rightarrow \Theta_1$ gives the state transition mapping that maps the transition in symbol sequence $\{\mathbf{x}_1\}$ from one state to another based on occurrence of a symbol in $\{\mathbf{x}_1\}$

5. ω_{12} is the symbol generation matrix of size $|\Theta_1| \times |\Xi_2|$; the ij^{th} entry of Ω_{12} denotes the probability of obtaining the symbol ξ_j of $\{\mathbf{x}_2\}$ while making a transition from the state θ_i of $\{\mathbf{x}_1\}$

Therefore, one can obtain the probability of a new symbol occurring after the previous D symbols are given for a particular sequence. Similarly, in order to find the probability of a new symbol occurring in a first symbol sequence with the last D symbols given of a second (different) symbol sequence, a xD-Markov machine can be applied correspondingly. Therefore, given a xD-Markov machine, the causal dependency of one symbol sequence on another symbol sequence can be captured.

2.8 Summary

This review has taken a bird's eye view approach to narrowing down, from the general ideas to specific details as to aid the architectures to be competitive in data science and be amongst the best performing algorithms.

CHAPTER 3. FEATURE EXTRACTION FROM SPATIAL DATA – LOW LIGHT NETWORK (LLNet)

In this chapter, hierarchical features that characterize basic 2-dimensional images are extracted for image preprocessing, enhancement and denoising tasks. However, the techniques can also be applied to multi-dimensional and multi-channel datasets with the only extra costs being to specify the number of channels or dimensions.

3.1 Introduction

There are recent improvements in image sensing techniques from 2-dimensional gray scale image acquisition to multi-channel RGB images. The most recent hyper-spectral imaging consists of a multiple of hundreds of wavelength bands. However, these images are almost corrupted by sources such as noise, low-light, smoke, fog, camera speed etc. Many applications acquire high quality images under low light conditions via pre-processing (either by algorithms, or changing a physical setup), post-processing, or using both. Hybrid camera systems have been designed to acquire high-quality images in low light conditions (Li (2011)) using flash photography techniques (e.g. near-infrared flash) in poorly illuminated environments combined with image enhancement algorithms (Matsui et al. (2010); Petschnigg et al. (2004); Eisemann and Durand (2004); Agrawal et al. (2005)). Some works suppressed noise artifacts (Chatterjee et al. (2011); Lee et al. (2005)), enhanced image contrast based on wavelet coefficients (Loza et al. (2013)) and used filters to reconstruct images due to photon noise detected under low-light conditions (Ford et al. (1997)). There are other well known enhancement methods such as improving image contrast by histogram equalization (Trahanias and Venetsanopoulos (1992); Cheng and Shi (2004); Pizer et al. (1987)) as well as denoising a variety of noise types using BM3D (Dabov et al. (2009, 2008, 2007)), K-SVD (Elad and Aharon (2006)) and non-linear filters (Chen et al. (1999); Chan et al. (2005)). For image denoising

task using deep learning, Vincent et al. (2008) presented the concept of denoising autoencoders for learning features from noisy images while Jain and Seung (2008) applied convolutional neural networks to denoise natural images. The network was applied for inpainting (Xie et al. (2012)) and deblurring (Schuler et al. (2014)). In addition, Agostinelli et al. (2013) implemented an adaptive multi-column architecture to robustly denoise images by training the model with various types of noise and testing on images with arbitrary noise levels and types. Stacked denoising autoencoders were used by Burger et al. (2012) to reconstruct clean images from noisy images by exploiting the encoding layer of the multilayer perceptron (MLP).

3.1.1 Motivation

In surveillance, monitoring and tactical reconnaissance, gathering visual information from a dynamic environment and accurately processing such data are essential to making informed decisions and ensuring the success of a mission. Camera sensors are often cost-limited to capture clear images or videos taken in a poorly-lit environment. Many applications aim to enhance brightness, contrast and reduce noise content from the images in an on-board real-time manner. A deep autoencoder-based approach is proposed to identify signal features from low-light images and adaptively brighten images without over-amplifying/saturating the lighter parts in images with a high dynamic range. We show that a variant of the stacked-sparse denoising autoencoder can learn to adaptively enhance and denoise from synthetically darkened and noise-added training examples. The model can be applied to images taken from natural low-light environment and/or are hardware-degraded. Results show significant credibility of the approach both visually and by quantitative comparison with various techniques. Good quality images and videos are important for automated and human-level decision-making for tasks ranging from security applications, military missions, path planning to medical diagnostics and commercial recommender systems. Clean, high-definition pictures captured by sophisticated camera systems provide better evidence for a well-informed course of action. However, cost constraints often limit large scale applications of such systems. Thus, relatively inexpensive sensors are used in many cases. Furthermore, adverse conditions such

as insufficient lighting (e.g. low-light environments, night time) worsen the situation. As a result, many areas of application, such as Intelligence, Surveillance and Reconnaissance (ISR) missions (e.g. recognizing and distinguishing enemy warships), unmanned vehicles (e.g. automated landing zones for UAVs), and commercial industries (e.g. property security, personal mobile devices) stand to benefit from improvements in image enhancement algorithms. The problem of contrast enhancement was approached from a learning perspective using deep autoencoders (what we refer to as Low-light Net, or LLNet) that are trained to learn underlying signal features in low-light images and adaptively brighten and denoise. The method utilized the advantage of the local patch-wise contrast improvement, similar to the works by Loza et al. (2013) to enhance contrast. The improvements are carried out relative to local neighbors to prevent over-amplifying the intensities of already brightened pixels. Furthermore, the same neural network is trained to learn the structures of objects that persist through noise in order to produce a brighter, denoised image. A novel application of using a class of deep neural networks – stacked sparse denoising autoencoder (SSDA) – to enhance natural low-light images is presented. A training data generation method is devised by synthetically modifying images available on Internet databases to simulate low-light environments. Two types of deep architecture are explored - (i) for simultaneous learning of contrast-enhancement and denoising (LLNet) and (ii) sequential learning of contrast-enhancement and denoising using two modules (*staged* LLNet or S-LLNet). The performances of the trained networks are evaluated and compared against other methods on test data with synthetic noise and artificial darkening. Performance evaluation was repeated on natural low-light images to demonstrate the enhancement capability of the synthetically trained model applied on a realistic set of images obtained with regular cell-phone camera in low-light environments. Hidden layer weights of the deep network were visualized to offer insights to the features learned by the model.

3.2 The Low-light Net (LLNet)

The proposed framework was introduced in this section along with training methodology and network parameters.

3.2.1 Learning features from low-light images with LLNet

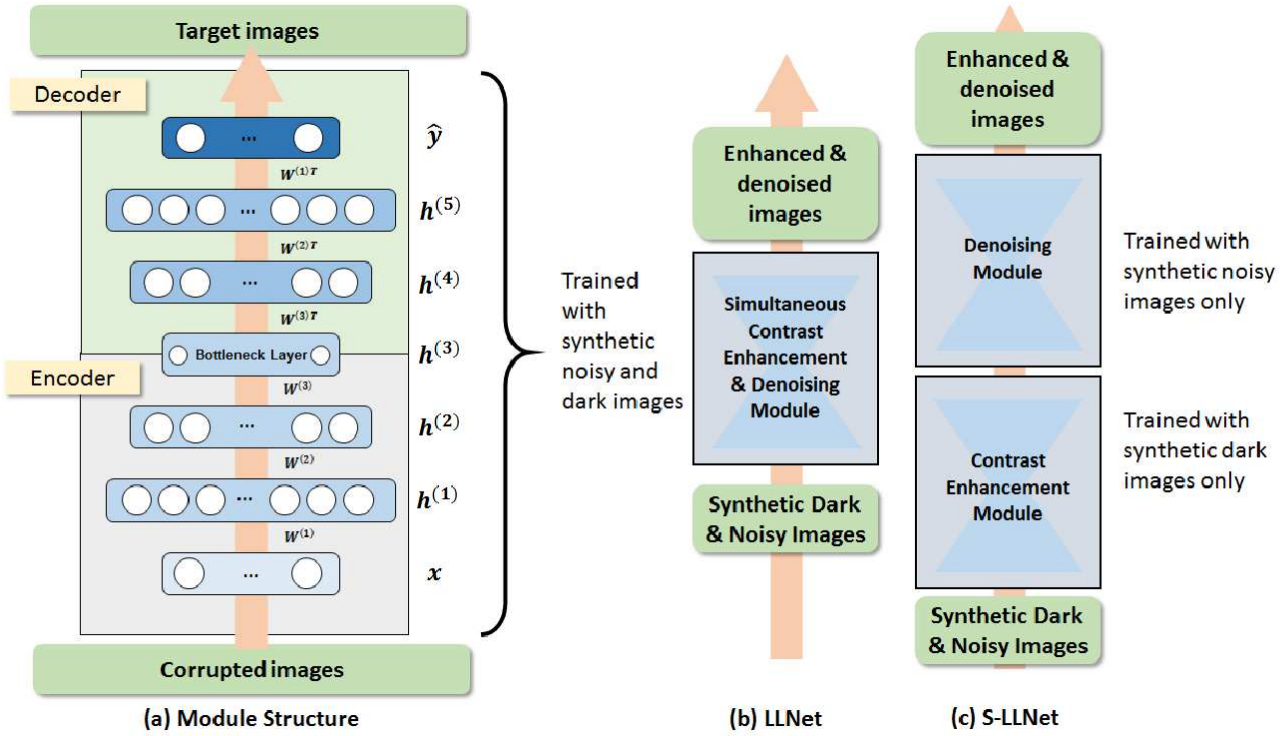


Figure 3.1: Architecture of the proposed framework: (a) An autoencoder module is comprised of multiple layers of hidden units, where the encoder is trained by unsupervised learning, the decoder weights are transposed from the encoder and subsequently fine-tuned by error back-propagation; (b) LLNet with a simultaneous contrast-enhancement and denoising module; (c) S-LLNet with sequential contrast-enhancement and denoising modules. The purpose of denoising is to remove noise artifacts often accompanying contrast enhancement.

LLNet framework takes its inspiration from SSDA Subsection 2.6.1 whose sparsity-inducing characteristic aids learning features to denoise signals. In the present work, we take the advantage of SSDA's denoising capability and the deep network's complex modeling capacity to learn features underlying in low-light images and produce enhanced images with minimal noise and improved contrast. A key aspect to be highlighted is that the network is trained using images obtained from internet databases that are subsequently synthetically processed (i.e. darkening nonlinearly and adding Gaussian noise) to simulate low-light conditions, since collection of a large number of

natural low-light images (sufficient for deep network training) and their well-lit counterparts can be unrealistic for practical use. Despite the fact that LLNet is trained on synthetic images, both synthetic and natural images are used to evaluate the network's performance in denoising and contrast-enhancement.

Aside from the regular LLNet where the network is trained with both darkened and noisy images, we also propose the *staged* LLNet (S-LLNet) which consists of separate modules arranged in series for contrast enhancement (stage 1) and denoising (stage 2). The key distinction over the regular LLNet is that the modules are trained separately with darkened-only training sets and noisy-only training sets. Both structures are presented in Figure 3.1. Note, while the S-LLNet architecture provides a greater flexibility of training (and certain performance improvement as shown in Section 3.4), it increases the inference time slightly which may be a concern for certain real-time applications. However, customized hardware-acceleration can resolve such issues significantly.

3.2.2 Network parameters

LLNet is comprised of 3 DA layers, with the first DA layer taking the input image of dimensions 17×17 pixels (i.e. 289 input units). The first DA layer has 867 hidden units, the second has 578 hidden units, and the third has 289 hidden units which becomes the bottleneck layer. Beyond the third DA layer forms the decoding counterparts of the first three layers, thus having 578 and 867 hidden units for the fourth and fifth layers respectively. Output units have the same dimension as the input, i.e. 289. The network was pre-trained for 30 epochs with pre-training learning rates of 0.1 for the first two DA layers and 0.01 for the last DA layer, whereas fine-tuning was performed with a learning rate of 0.1 for the first 200 fine-tuning epochs, 0.01 afterwards, and stops only if the improvement in validation error is less than 0.5%. For the case of S-LLNet, the parameters of each module are identical.

3.2.3 Training the model

Training was performed using 422,500 patches, extracted from 169 standard test images[1]. Consistent with current practices, the only pre-processing done was to normalize the image pixels to between zero and one. During the generation of the patches, we produced 2500 patches from random locations (and with random darkening and noise parameters) from the same image. The 17×17 pixel patches are then darkened nonlinearly using the *MatLab* command `imadjust` to randomly apply a gamma adjustment. Gamma correction is a simple but general case with application of a power law formula to images for pixel-wise enhancement with the following expression:

$$I_{\text{out}} = A \times I_{\text{in}}^{\gamma} \tag{3.1}$$

where A is a constant determined by the maximum pixel intensity in the image. Intuitively, image is brightened when $\gamma < 1$ while $\gamma = 1$ leaves it unaffected. Therefore, when $\gamma > 1$, the mapping is weighted toward lower (darker) gray-scale pixel intensity values.

A uniform distribution of $\gamma \sim$ Uniform $(2, 5)$ with random variable γ, is selected to result in training patches that are darkened to a varying degree, thus simulating multiple low-light scenarios possible in real-life. Additionally, to simulate low quality cameras used to capture images, these training patches are further corrupted by a Gaussian noise via *MatLab* function `imnoise` with standard deviation of $\sigma = \sqrt{B(25/255)^2}$, where $B \sim$ Uniform $(0, 1)$. Hence, the final corrupted image and the original image exhibit the following relationship:

$$I_{\text{train}} = n\left(g(I_{\text{original}})\right) \tag{3.2}$$

where function $g(\cdot)$ represents the gamma adjustment function and $n(\cdot)$ represents the noise function.

Random gamma darkening with random noise levels result in a variety of training images that can increase the robustness of the model. In reality, natural low-light images may also include quantization and Poisson noise in addition to Gaussian noise. We chose a Gaussian-only model for the ease of analysis and as a preliminary feasibility study of the framework trained on synthetic

[1]Dataset URL: http://decsai.ugr.es/cvg/dbimagenes/

images and applied to natural images. Furthermore, since Gaussian noise is a very familiar yet popular noise model for many image denoising tasks, we can acquire a sense of how well LLNet performs with respect to other image enhancement algorithms. The patches are randomly shuffled and then divided into 211,250 training examples and 211,250 validation samples. The training step involves learning the invariant representation of low light and noise with the autoencoder described in Section 3.2.2. While training the model, the network attempts to remove the noise and simultaneously enhance the contrast of these darkened patches. The reconstructed image is compared against the clean version (i.e. bright, noiseless image) by computing the mean-squared error.

When training both LLNet and S-LLNet, each DA is trained by error back-propagation to minimize the sparsity regularized reconstruction loss as described by Xie et al. (2012):

$$\mathcal{L}_{\mathrm{DA}}(\mathcal{D};\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||y_i - \hat{y}(x_i)||_2^2 + \beta\sum_{j=1}^{K}\mathrm{KL}(\hat{\rho}_j||\rho) + \frac{\lambda}{2}(||W||_F^2 + ||W'||_F^2) \tag{3.3}$$

where N is the number of patches, $\theta = \{W, b, W', b'\}$ are the parameters of the model, $\mathrm{KL}(\hat{\rho}_j||\rho)$ is the Kullback-Leibler divergence between ρ (target activation) and $\hat{\rho}_j$ (empirical average activation of the j-th hidden unit) which induces sparsity in the hidden layers:

$$\mathrm{KL}(\hat{\rho}_j||\rho) = \rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \tag{3.4}$$

where

$$\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N} h_j(x_i) \tag{3.5}$$

and λ, β and ρ are scalar hyper-parameters determined by cross-validation.

After the weights of the decoder have been initialized, the entire pre-trained network is fine-tuned using an error back-propagation algorithm to minimize the loss function given by:

$$\mathcal{L}_{\mathrm{SSDA}}(\mathcal{D};\theta) = \frac{1}{N}\sum_{i=1}^{N}||y_i - \hat{y}(x_i)||_2^2 + \frac{\lambda}{L}\sum_{l=1}^{2L}||W^{(l)}||_F^2$$

where L is the number of stacked DAs and $W^{(l)}$ denotes weights for the l-th layer in the stacked deep network. The sparsity inducing term is not needed for this step because the sparsity was already incorporated in the pre-trained DAs.

3.2.4 Image reconstruction

During inference, the test image was first broken up into overlapping 17×17 patches with stride size of 3×3. The collection of patches was then passed through LLNet to obtain corresponding denoised, contrast-enhanced patches. The patches were averaged and re-arranged back into its original dimensions. From our experiments, it was discovered that using a patching stride of 2×2 or even 1×1 (fully overlapped patches) do not produce significantly superior results. Additionally, increasing the number of DA layers improves the nonlinear modeling capacity of the network. However, a larger model was more computationally expensive to train and the network structure was determined to be adequate for the current study.

3.3 Evaluation metrics and compared methods

In this section, a brief descriptions of other contrast-enhancement methods along with the performance metric used to evaluate the proposed framework's performance.

3.3.1 Performance metric

Two metrics are used, namely the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM).

3.3.1.1 Peak signal-to-noise ratio (PSNR)

PSNR quantifies the extent of corruption of original image with noise as well as approximating human perception of the image. It has also been established to demonstrate direct relationship with compression-introduced noise (Santoso et al. (2011)). Roughly, the higher the PSNR, the better the denoised image especially with the same compression code. Basically, it is a modification of the mean squared error between the original image and the reconstructed image. Given a noise-free

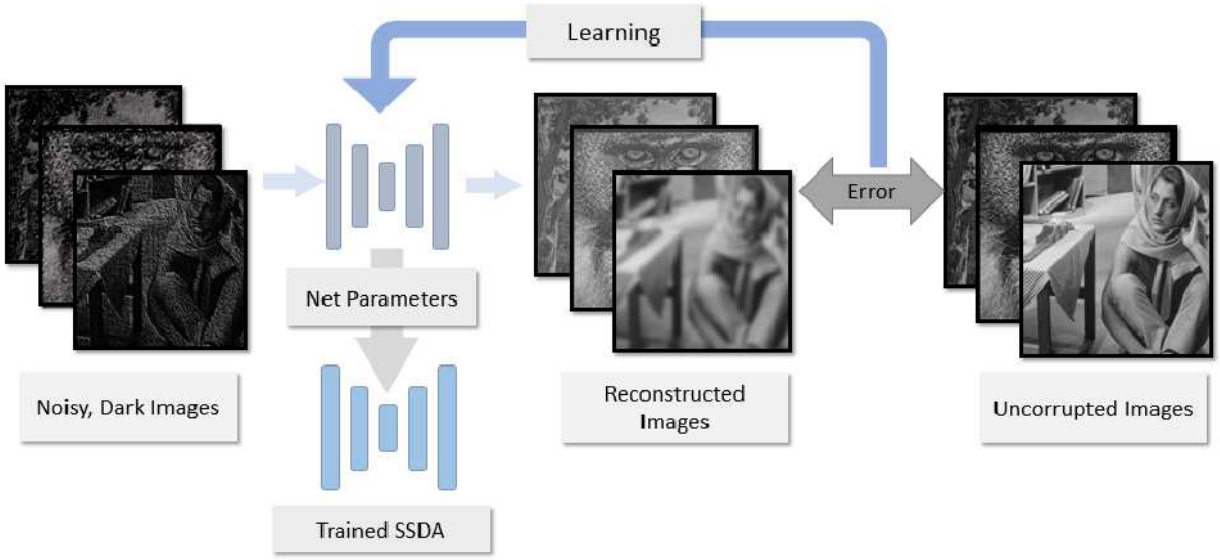


Figure 3.2: Training the LLNet: Training images are synthetically darkened and added with noise. These images are fed through LLNet where the reconstructed images are compared with the uncorrupted images to compute the error, which is then back-propagated to finetune and optimize the model weights and biases.

$m \times n$ monochrome image I and its reconstructed version K, MSE is expressed as:

$$\text{MSE} = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3.6}$$

The PSNR, in decibels (dB) is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\max(I)^2}{\text{MSE}} \right) \tag{3.7}$$

Here, $\max(I)$ is the maximum possible pixel value of the image I.

3.3.1.2 Structural similarity index (SSIM)

SSIM is a metric for capturing the perceived quality of digital images and videos (Loza et al. (2013); Wang et al. (2004)). It is used to measure the similarity between two images. SSIM quantifies the measurement or prediction of image quality with respect to initial uncompressed or distortion-free image as reference. As PSNR and MSE are known to quantify the absolute error between the result and the reference image, such metrics may not really quantify complete similarity.

On the other hand, SSIM explores the change in image structure and being a perception-type model, it incorporates pixel inter-dependencies as well as masking of contrast and pixel intensities. SSIM is expressed as:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.8}$$

where μ_x is the average of window x, μ_y is the average of window y, σ_x^2 is the variance of x, σ_y^2 is the variance of y, σ_{xy}^2 is the covariance of x and y, $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator with $k_1 = 0.01$ and $k_2 = 0.03$ by default, and L is the dynamic range of pixel values.

3.3.2 Compared methods

This subsection describes certain popular methods for enhancing low-light images used here for comparison.

3.3.2.1 Histogram equalization (HE)

Histogram equalization usually increases the global contrast of images, especially when the usable data of the image is represented by close contrast values (Trahanias and Venetsanopoulos (1992); Cheng and Shi (2004); Pizer et al. (1987)). Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed.

3.3.2.2 Contrast-limiting adaptive histogram equalization (CLAHE)

Contrast-limiting adaptive histogram equalization differs from ordinary adaptive histogram equalization in its contrast limiting. In the case of CLAHE, the contrast limiting procedure has to

be applied for each neighborhood from which a transformation function is derived (Pisano et al. (1998)). CLAHE was developed to prevent the over-amplification of noise that arise in adaptive histogram equalization.

3.3.2.3 Gamma adjustment (GA)

The simple form of gamma correction is outlined in Equation (3.2). Gamma curves illustrated with $\gamma > 1$ have exactly the opposite effect as those generated with $\gamma < 1$. It is important to note that gamma correction reduces toward the identity curve when $\gamma = 1$. As discussed earlier in Section 3.2.3, the image is brightened when $\gamma < 1$, darkened with $\gamma > 1$, while $\gamma = 1$ leaves it unaffected.

3.3.2.4 Histogram equalization with 3D block matching (HE+BM3D)

BM3D is the current state-of-the-art algorithm for image noise removal presented by Dabov et al. (2007). It uses a collaborative form of Wiener filter for high dimensional block of patches by grouping similar 2D blocks into a 3D data array. The algorithm ensures the sparsity in transformed domain and takes advantage of joint denoising of grouped patches similar in ways to pixel-wise overcompleteness which K-Singular value Decomposition (KSVD) (Elad and Aharon (2006)), the former best performing denoising method), ensured on patch-based dictionaries. Finally, domain inversion is done and the results of different matched block are fused together. In this work we attempt to first equalize the contrast of the test image, then use BM3D as a denoiser to remove the noise resulting from histogram equalization. Attempt was made to reverse the order, i.e. use BM3D to remove noise from the low-light images first then apply contrast enhancement. As BM3D removes noise by patching images, the patch boundaries get significantly amplified and become extremely pronounced when histogram equalization is applied, hence, producing non-competitive results.

3.4 Results and Discussion

In this section, we evaluate the performance of our framework against the methods outlined above on standard images shown in Figure 3.3. Test images are darkened with $\gamma = 3$, where noisy versions contain Gaussian noise of $\sigma = 18$ and $\sigma = 25$, which are typical values for image noise under poor illumination and/or high temperature; these parameters correspond to scaled variances of $\sigma_s^2 = 0.005$ and $\sigma_s^2 = 0.010$ respectively if the pixel intensities are in 8-bit integers ($\sigma_s = \sigma/255$ where $\sigma_s \in [0, 1]$ and $\sigma \in [0, 255]$). Histogram equalization is performed by using the *MatLab* function `histeq`, whereas CLAHE was performed with the function `adapthisteq` with default parameters (8×8 image tiles, contrast enhancement limit of 0.01, full range output, 256 bins for building contrast enhancing transformation, uniform histogram distribution, and distribution parameter of 0.4). Gamma adjustment is performed on the test image with $\gamma = 0.3$. For the hybrid 'HE+BM3D' method, histogram equalization was first applied to enhance image contrast before using the BM3D code developed by Dabov et al. (2007) as a denoiser, where the noise standard deviation input parameter for BM3D was set to $\sigma = 25$ (the highest noise level of the test image). Both LLNet and S-LLNet outputs were reconstructed with overlapping 17×17 patches of stride size 3×3. Training was performed on NVIDIA's TITAN X GPU using Theano's deep learning framework (Bastien et al. (2012); Bergstra et al. (2010a)) and took approximately 30 hours. Enhancing an image with dimension of 512×512 pixels took 0.42 s on GPU.

3.4.1 Algorithm adaptivity

Ideally, an already-bright image should no longer be brightened any further. To test this, different enhancement algorithms are performed on a normal, non-dark and noiseless image. Figure 3.4A shows the result when running the 'Town' image through various algorithms. LLNet outputs a slightly brighter image, but not to the degree that everything appears over-brightened and washed-out like that in GA output. This shows that in the process of learning low-light features, LLNet successfully learns the necessary degree of required brightening that should be applied to the image. However, when evaluating contrast enhancement via visual inspection, histogram equal-

Table 3.1: PSNR and SSIM of outputs using different enhancement methods. 'Bird' means the non-dark and noiseless (i.e. original) image of Bird. 'Bird-D' indicates a darkened version of the same image. 'Bird-D+GN18' denotes a darkened Bird image with added Gaussian noise of $\sigma = 18$, whereas 'Bird-D+GN25' denotes darkened Bird image with added Gaussian noise of $\sigma = 25$. Bolded numbers corresponds to the method with the highest PSNR or SSIM. Asterisk (*) denotes our framework.

SSIM (dB) / SSIM	**Dark**	**HE**	**CLAHE**	**GA**	**HE+BM3D**	**LLNet***	**S-LLNet***
Bird	N/A	9.56 / 0.48	**19.77 / 0.82**	8.68 / 0.48	9.58 / 0.50	15.89 / 0.69	14.38 / 0.63
Bird-D	14.80 / 0.27	9.57 / 0.44	19.09 / 0.72	19.22 / 0.75	9.61 / 0.48	**24.57 / 0.77**	19.97 / 0.61
Bird-D+GN18	14.93 / 0.20	7.71 / 0.07	15.47 / 0.12	12.72 / 0.10	8.21 / 0.10	**24.76** / 0.58	23.71 / **0.64**
Bird-D+GN25	14.96 / 0.16	7.53 / 0.07	13.85 / 0.09	11.80 / 0.08	8.00 / 0.09	23.02 / 0.46	**24.72 / 0.60**
Girl	N/A	18.06 / 0.81	16.88 / 0.71	10.85 / 0.78	18.03 / 0.68	**19.05 / 0.83**	15.73 / 0.76
Girl-D	9.58 / 0.53	18.09 / 0.81	13.74 / 0.66	**30.08 / 0.99**	18.06 / 0.68	20.83 / 0.79	21.32 / 0.73
Girl-D+GN18	8.79 / 0.20	15.88 / 0.26	12.42 / 0.17	16.82 / 0.30	18.87 / 0.53	18.40 / 0.59	**20.59 / 0.65**
Girl-D+GN25	8.76 / 0.15	15.14 / 0.19	11.61 / 0.13	15.01 / 0.20	18.11 / 0.39	17.98 / 0.48	**20.72 / 0.59**
House	N/A	13.36 / 0.70	**18.89 / 0.81**	9.50 / 0.56	13.24 / 0.61	11.61 / 0.60	10.57 / 0.50
House-D	12.12 / 0.33	12.03 / 0.65	16.81 / 0.60	**26.79 / 0.82**	11.92 / 0.54	21.10 / 0.64	18.73 / 0.49
House-D+GN18	12.19 / 0.29	10.55 / 0.33	15.48 / 0.35	13.76 / 0.33	11.39 / 0.42	**20.25 / 0.56**	19.91 / 0.51
House-D+GN25	12.16 / 0.26	10.09 / 0.29	14.08 / 0.29	12.67 / 0.28	10.94 / 0.37	19.71 / **0.52**	**20.76** / 0.51
Pepper	N/A	**23.26 / 0.92**	19.21 / 0.79	10.07 / 0.70	23.10 / 0.83	12.27 / 0.72	11.12 / 0.68
Pepper-D	10.57 / 0.42	22.83 / 0.87	14.25 / 0.59	**28.23 / 0.89**	22.73 / 0.80	20.48 / 0.76	19.35 / 0.70
Pepper-D+GN18	9.71 / 0.18	15.81 / 0.22	13.42 / 0.17	14.73 / 0.22	19.02 / 0.58	19.40 / 0.55	**20.03 / 0.63**
Pepper-D+GN25	9.08 / 0.14	14.43 / 0.16	12.39 / 0.13	13.62 / 0.16	17.29 / 0.37	18.38 / 0.47	**20.55 / 0.60**
Town	N/A	18.88 / 0.84	16.4 / 0.74	9.47 / 0.71	**18.95** / 0.78	17.82 / **0.87**	16.64 / 0.81
Town-D	11.14 / 0.43	18.89 / 0.84	15.86 / 0.70	22.71 / **0.94**	18.98 / 0.78	**24.43** / 0.80	21.83 / 0.69
Town-D+GN18	10.07 / 0.18	14.86 / 0.26	13.22 / 0.19	15.03 / 0.25	17.51 / 0.41	20.25 / 0.60	**21.78 / 0.60**
Town-D+GN25	10.08 / 0.14	14.05 / 0.21	12.23 / 0.14	13.61 / 0.18	16.32 / 0.32	19.76 / 0.50	**23.53 / 0.58**
Anhinga	N/A	13.69 / 0.67	**22.32 / 0.92**	8.76 / 0.63	13.76 / 0.67	13.72 / 0.75	10.51 / 0.61
Anhinga-D	12.65 / 0.37	13.70 / 0.68	16.83 / 0.72	19.58 / **0.91**	13.76 / 0.67	**22.26** / 0.78	18.73 / 0.59
Anhinga-D+GN18	12.68 / 0.27	11.93 / 0.25	15.05 / 0.28	14.09 / 0.29	13.05 / 0.29	**22.79** / 0.65	22.72 / **0.66**
Anhinga-D+GN25	12.69 / 0.24	11.51 / 0.22	13.70 / 0.23	13.04 / 0.24	12.58 / 0.26	22.30 / 0.56	**22.66 / 0.61**
Avion	N/A	10.50 / 0.56	**19.70** / 0.83	14.41 / 0.83	10.54 / 0.63	16.06 / **0.85**	15.77 / 0.84
Avion-D	12.90 / 0.78	10.50 / 0.56	14.03 / 0.72	27.59 / **0.99**	10.54 / 0.63	**28.22** / 0.96	27.17 / 0.93
Avion-D+GN18	9.52 / 0.29	10.24 / 0.22	9.50 / 0.22	19.61 / 0.53	10.83 / 0.42	21.99 / 0.75	**22.44 / 0.80**
Avion-D+GN25	8.78 / 0.23	10.11 / 0.19	8.89 / 0.19	17.79 / 0.41	10.77 / 0.31	19.50 / 0.63	**21.32 / 0.73**
Baboon	N/A	17.69 / **0.82**	17.24 / 0.77	10.94 / 0.73	**18.05** / 0.78	12.10 / 0.72	12.43 / 0.63
Baboon-D	9.48 / 0.57	17.68 / 0.82	14.03 / 0.72	**29.88 / 0.99**	18.05 / 0.78	21.94 / 0.85	21.00 / 0.68
Baboon-D+GN18	9.01 / 0.35	15.34 / 0.46	12.61 / 0.40	16.01 / 0.48	17.59 / 0.62	18.03 / 0.68	**21.03 / 0.62**
Baboon-D+GN25	8.02 / 0.26	14.51 / 0.38	11.61 / 0.32	14.42 / 0.38	16.67 / 0.50	15.83 / **0.58**	**19.96** / 0.56
Barnfall	N/A	13.16 / 0.73	**19.27 / 0.85**	8.62 / 0.66	13.33 / 0.69	14.16 / 0.76	13.78 / 0.67
Barnfall-D	11.48 / 0.31	13.17 / 0.72	15.66 / 0.68	**21.48 / 0.93**	13.35 / 0.69	21.07 / 0.63	18.11 / 0.44
Barnfall-D+GN18	10.93 / 0.15	10.91 / 0.16	13.79 / 0.17	13.63 / 0.17	11.92 / 0.21	20.02 / **0.46**	**20.61** / 0.43
Barnfall-D+GN25	10.81 / 0.12	10.44 / 0.12	12.66 / 0.12	12.61 / 0.12	11.37 / 0.16	19.71 / 0.40	**21.62 / 0.42**
Beeflowr	N/A	14.85 / 0.76	**23.22 / 0.93**	9.49 / 0.60	14.84 / 0.71	14.79 / 0.73	14.28 / 0.67
Beeflowr-D	13.77 / 0.47	14.93 / 0.74	18.24 / 0.82	17.32 / 0.79	14.97 / 0.70	**27.20 / 0.82**	22.95 / 0.73
Beeflowr-D+GN18	12.45 / 0.21	12.28 / 0.17	14.84 / 0.14	13.67 / 0.19	13.56 / 0.41	24.24 / 0.57	**24.41 / 0.65**
Beeflowr-D+GN25	12.24 / 0.16	11.61 / 0.13	13.34 / 0.10	12.72 / 0.13	12.85 / 0.32	21.03 / 0.43	**22.43 / 0.54**
Blakeyed	N/A	13.94 / 0.62	**23.14 / 0.80**	11.08 / 0.66	13.88 / 0.51	12.41 / 0.54	11.75 / 0.47
Blakeyed-D	11.60 / 0.51	11.56 / 0.59	16.28 / 0.63	**28.75 / 0.86**	11.52 / 0.48	25.06 / 0.65	19.75 / 0.53
Blakeyed-D+GN18	11.77 / 0.21	11.46 / 0.22	14.62 / 0.19	13.26 / 0.23	12.50 / 0.39	22.12 / 0.46	**22.45 / 0.51**
Blakeyed-D+GN25	11.18 / 0.16	10.81 / 0.17	13.24 / 0.14	12.40 / 0.17	11.91 / 0.31	19.70 / 0.38	**20.55 / 0.44**
Winning Instances	N/A	1 / 2	7 / 6	6 / 9	2 / 0	9 / 9	**19 / 18**
Total Instances	N/A	44	44	44	44	44	44

Figure 3.3: Original standard test images used to compute PSNR.

ization methods (i.e. HE, CLAHE, HE+BM3D) provide superior enhancement given the original image. When tested with other images (namely, 'Bird', 'Girl', 'House', 'Pepper', etc.) as shown in Table 3.1, HE-based methods generally fared slightly better with higher PSNR and SSIM.

3.4.2 Enhancing artificially darkened images

Figure 3.4B shows output of various methods when enhancement is applied to a 'Town' image darkened with $\gamma = 3$. Here, LLNet achieves the highest PSNR followed by GA, but the the other way round when evaluated with SSIM. The similarity of GA-enhanced image with the original is expected because gamma readjustment with $\gamma = 0.3 \simeq 1/3$ essentially reverses the process close to the original intensity levels. In fact, when tested with other images, the highest scores for darkened-only images are achieved only by one of LLNet, S-LLNet or GA. Note that LLNet is trained on varying degrees of γ but not with a fixed $\gamma = 3$. Results tabulated in Table 3.1 highlights the advantages and broad applicability of the deep autoencoder approach with LLNet and S-LLNet.

3.4.3 Enhancing darkened images in the presence of synthetic noise

To simulate dark images taken with regular or subpar camera sensors, Gaussian noise is added to the synthetic dark images. Figure 3.4C and 3.4D presents a gamma-darkened 'Town' image

Figure 3.4: Comparison of methods of enhancing 'Town' when applied to (A) original already-bright, (B) darkened, (C) darkened and noisy ($\sigma = 18$), and (D) darkened and noisy ($\sigma = 25$) images. Darkening is done with $\gamma = 3$. The numbers with units dB are PSNR, the numbers without are SSIM.

corrupted with Gaussian noise of $\sigma = 18$ and $\sigma = 25$, respectively. For this test image, S-LLNet attains the highest PSNR and SSIM followed by LLNet for both noise levels. Table 3.1 shows that no other methods aside from LLNet/S-LLNet result in the highest PSNR and SSIM for dark, noisy images. Histogram equalization methods fail due to the intensity of noisy pixels being equalized and produced detrimental effects to the output images. Additionally, BM3D is not able to effectively denoise the equalized images with parameter $\sigma = 25$ since the structure of the noise changes during the equalization process.

3.4.4 Application on natural low-light images

When working with downloaded images, a clean reference image is available for computing PSNR and SSIM. However, reference images may not be available in real life when working with naturally dark images. The experiment was controlled to circumvent the issue by mounting an ordinary cell-phone (Nexus 4) camera on a tripod to capture pictures in an indoor environment with both lights on and lights off. The picture with lights on are used as the reference images for PSNR and SSIM computations, whereas the picture with lights off becomes the natural low-light test image. Although the bright pictures cannot be considered as the ground truth, it provides a reference point to evaluate the performance of various algorithms. Performance of each enhancement method is shown in Figure 3.5. While histogram equalization greatly improves the contrast of the image, it corrupts the output with a large noise content. In addition, the method suffers from over-amplification in regions where there is a very high intensity brightness in dark regions, as shown by blooming effect on the computer display in panel 3.5A(vi) and 3.5A(vii). CLAHE is able to improve the contrast without significant blooming of the display, but like HE it tends to amplify noise within the images. LLNet performs significantly well with its capability to suppress noise in most of the images while improving local contrast, as shown in the magnified patches at the bottom of Figure 3.5.

3.4.5 Denoising capability, image sharpness, and patch size

There is a trade-off between denoising capability and the perceived sharpness of the enhanced image. While higher PSNR indicates a higher denoising capability, this metric favors smoother edges. Therefore, images that are less sharp often achieve a higher PSNR. Hence, SSIM is used as a complementary metric to evaluate the gain or loss in perceived structural information. From the experiments, a relationship between denoising capability (PSNR), similarity levels (SSIM) and image sharpness is found to be dependent on the dimensions of the denoised patch relative to the test image. A smaller patch size implies finer-grain enhancement over the test image, whereas a larger patch size implies coarser enhancement. Because natural images may also come in varying

heights and widths, the relative patch size–a dimensionless quantity that relates the patch size to the dimensions of the test image, r–is defined as:

$$r = \frac{d_p}{d_i} = \frac{\sqrt{w_p^2 + h_p^2}}{\sqrt{w_i^2 + h_i^2}}$$

where quantities d, w, and h denote the diagonal length, width, and height in pixels, with subscripts p and i referring to the patch and test image respectively. Relative patch size may also be thought as the size of the receptive field on a test image. From the results, it is observed that when the relative patch size decreases, object edges appear sharper at the cost of having more noise. However, there exists an optimal patch size resulting in an enhanced image with the highest PSNR or SSIM (as shown in Figure 3.6 and Figure 3.7.). If the optimal patch size is selected based on PSNR, the resulting image will have the lowest noise levels but is less sharp. If the smallest patch size is selected, then the resulting image has the highest sharpness where more details can be observed but with the expense of having more noise. Choosing the optimal patch size based on SSIM produces a more well-balanced result in terms of denoising capability and image sharpness.

We included a natural test image where the US Air Force (USAF) resolution test chart is shown. The test chart consists of groups of three bars varying in sizes labeled with numbers which conforms to the MIL-STD-150A standard set by the US Air Force in 1951. Originally, this test chart is used to determine the resolving power of optical imaging systems such as microscopes, cameras, and image scanners. For the present study, we used this test chart to visually compare the trade-off denoising capability and image sharpness using different relative patch sizes. The results are shown in Figure 3.7.

3.4.6 Prior knowledge on input

HE can be easily performed on images without any input parameters. Like HE, CLAHE can also be used without any input parameters where the performance can be further finetuned with various other parameters such as tile sizes, contrast output ranges, etc. Gamma adjustment and BM3D both require prior knowledge of the input parameter (values of γ and σ, respectively), thus

it is often necessary to finetune the parameters by trial-and-error to achieve the best results. The advantages of using deep learning-based approach, specifically using LLNet and S-LLNet, is that after training them with a large variety of (darkened and noisy) images (with proper choice of hyper-parameters), there is no need for meticulous hand-tuning during testing/practical use. The model automatically extracts and learns the underlying features from low-light images. Essentially, this study shows that a deep model that has been trained with varying degrees of darkening and noise levels can be used for many real-world problems without detail knowledge of camera and environment.

3.4.7 Features of low-light images

To gain an understanding on what features are learned by the model, weights linking the input to the first layer of the trained model can be visualized by plotting the values of the weight matrix as pixel intensity values (Figure 3.8). In a regular LLNet where both contrast enhancement and denoising are learned simultaneously, the weights contain blob-like structures with prominent coarse-looking textures. Decoupling the learning process (in the case of S-LLNet) allows us to acquire a better insight. We observe that blob-like structures are learned when the model is trained for the task of contrast enhancement. The shape of the features suggest that contrast enhancement considered localized features into account; if a region is dark, then the model brightens it based on the context in the patch (i.e. whether the edge of an object is present or not). On the other hand, feature detectors for the denoising task appears noise-like, albeit in a finer-looking texture compared to the on coarser ones from the integrated network, LLNet. These features shows that the denoising task is mostly performed in an overall manner. Note that while the visualizations presented in Burger et al. (2012) show prominent Gabor-like features at different orientations for the denoising task, the Gabor-like features are not apparent in the present study because the training data consists of multiple noise levels rather than a fixed one. Due to specialization in different tasks, it is not surprising that S-LLNet achieves superior performance over LLNet at higher noise

Table 3.2: Average PSNR evaluated on the set of natural dark images enhanced with trained model of different hyper-parameters and network architecture. The baseline model is marked by an asterisk (*), whereas the PSNR for the best model is presented in a bolded typeface. 'Btk' is an abbreviation of 'bottleneck' that describes a model that has a decreasing number of hidden units toward the center.

Model Description	Network Architecture (# Hidden Units)	PSNR (dB)	SSIM
*Batch Size 50	867-578-289-578-867	**16.1307**	0.6560
Batch Size 500	867-578-289-578-867	16.1101	**0.6603**
Batch Size 1000	867-578-289-578-867	15.8714	0.6514
Patch Size 13×13	867-578-289-578-867	16.6254	**0.6913**
*Patch Size 17×17	867-578-289-578-867	16.1307	0.6560
Patch Size 21×21	867-578-289-578-867	**17.3355**	0.6724
Patch Size 25×25	867-578-289-578-867	16.0448	0.6660
3-layer SDA	578-289-578	15.6375	0.6486
*5-layer SDA	867-578-289-578-867	16.1307	**0.6560**
7-layer SDA	1156-867-578-289-578-867-1156	16.3170	0.6551
9-layer SDA	1445-1156-867-578-289-578-867-1156-1445	**16.3463**	0.6380
*Regular, w. Btk.	867-578-289-578-867	16.1307	0.6560
Narrow, w/o. Btk.	289-289-289-289-289	**16.6713**	**0.6619**
Regular, w/o. Btk.	578-578-578-578-578	15.7735	0.6557
Wide, w/o. Btk.	867-867-867-867-867	15.2639	0.6526
Narrow	578-289-145-289-578	**16.2796**	**0.6604**
*Regular	867-578-289-578-867	16.1307	0.6560
Wide	1156-867-578-867-1156	15.8524	0.6581

levels. The distinction between feature detectors and feature generators is highlighted in Figure 3.9 and a comparison of superior and inferior weights is shown in Figure 3.10.

3.4.8 Hyper-parameters, network architecture, and performance

Table 3.2 shows the average PSNR and SSIM values evaluated on the set of natural dark images enhanced with the trained model of different hyper-parameters and network architecture. Note that these are merely the results of exploring the hyper-parameter space to determine how the selection of hyper-parameters affect the image enhancement performance. The optimal hyper-parameters are not implemented in the LLNet and S-LLNet models reported above (see Section 3.2.3 for implemented values) and hence there is further room for improvement. From the results, a relatively smaller training batch size results in higher PSNR, but no clear trend is observed in terms of SSIM. Smaller batch sizes result in noisier gradient during the update and may help in escaping local minima during optimization. A patch size of 17×17 resulted in the highest average

PSNR whereas a patch size of 13x13 resulted in the highest SSIM. This result has been discussed in earlier sections of this chapter and is consistent with the findings on the relationship between the relative patch size and PSNR and SSIM. On the other hand, a deeper and narrower network performed better than shallow and wider network. It has also been found that a bottlenecked network produces higher PSNR and SSIM than a model with uniform number of hidden units across layers. The bottleneck region helps in extracting better features during the training phase and corroborates the findings by Gehring et al. (2013).

3.5 Summary

Feature extraction and data preprocessing (corruption removal) from spatial data has been attacked using a designed deep learning technique called low light network (LLNet). LLNet is a variant of the stacked sparse denoising autoencoder that was trained to learn the brightening and denoising functions from various synthetic examples as filters which are then applied to enhance naturally low-light and degraded images. Results show that deep learning based approaches are suitable for such tasks for natural low-light images of varying degree of degradation. The proposed LLNet (and S-LLNet) compete favorably with currently used image enhancement methods such as histogram equalization, CLAHE, gamma adjustment, and hybrid methods such as applying HE first and subsequently using a state of the art denoiser (i.e.,BM3D).

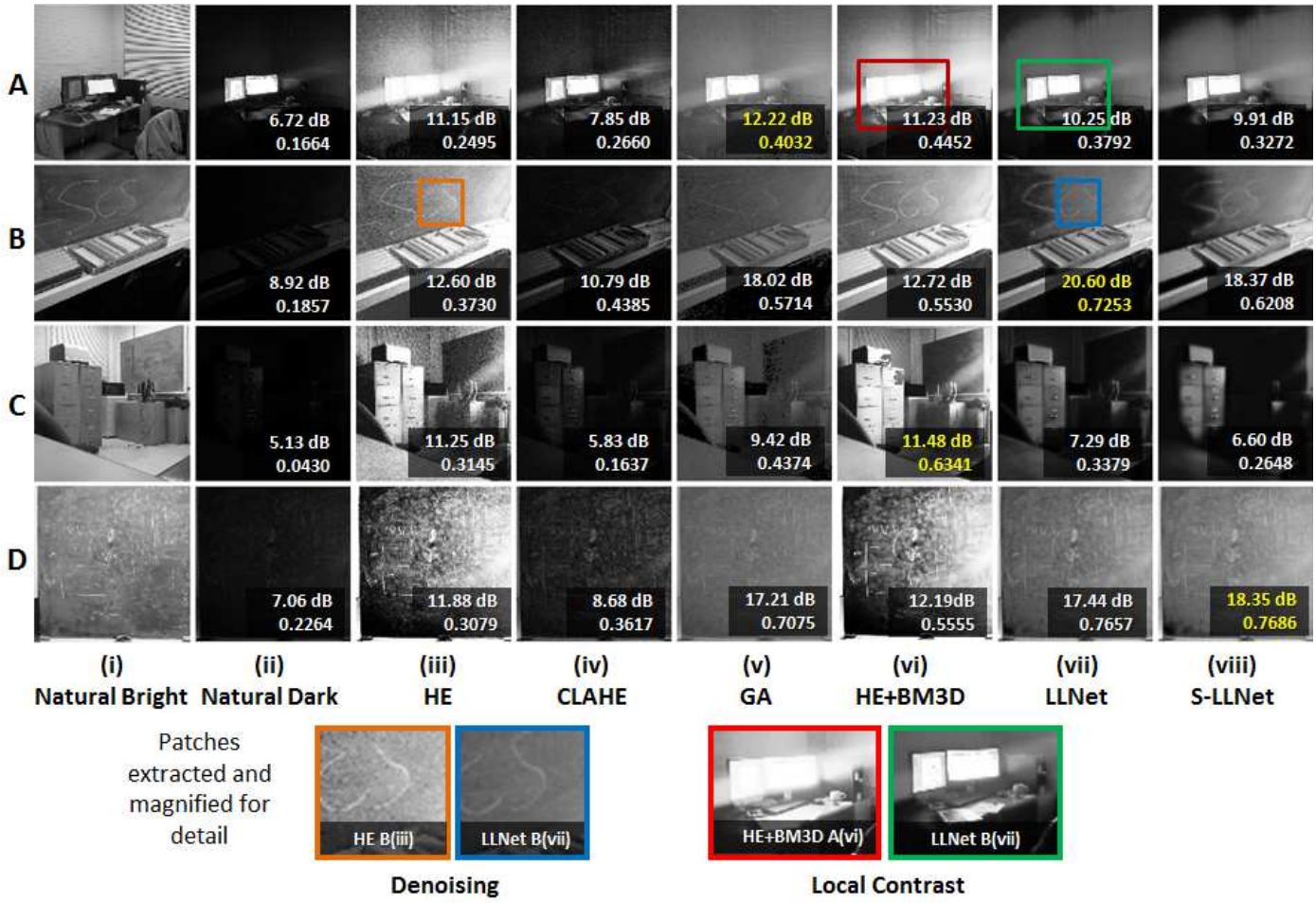


Figure 3.5: Comparison of methods of enhancing naturally dark images of (A) computer, (B) chalkboard, (C) cabinet, and (D) writings. Selected regions are enlarged to demonstrate the denoising and local contrast enhancement capabilities of LLNet. HE (including HE+BM3D) results in over-amplification of the light from the computer display whereas LLNet was able to avoid this issue. In the natural case, the best performers in terms of PSNR and SSIM coincide.

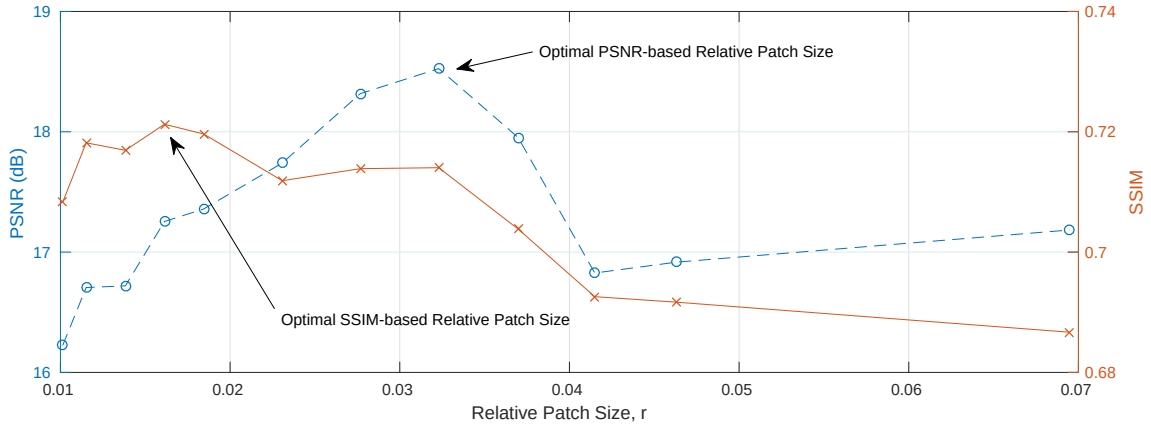


Figure 3.6: Relative patch size vs PSNR and SSIM. The picture with highest PSNR has the highest denoising capability but least sharp. Picture with lowest r has the least denoising capability but has the highest image sharpness. Picture with the highest SSIM is more well-balanced.

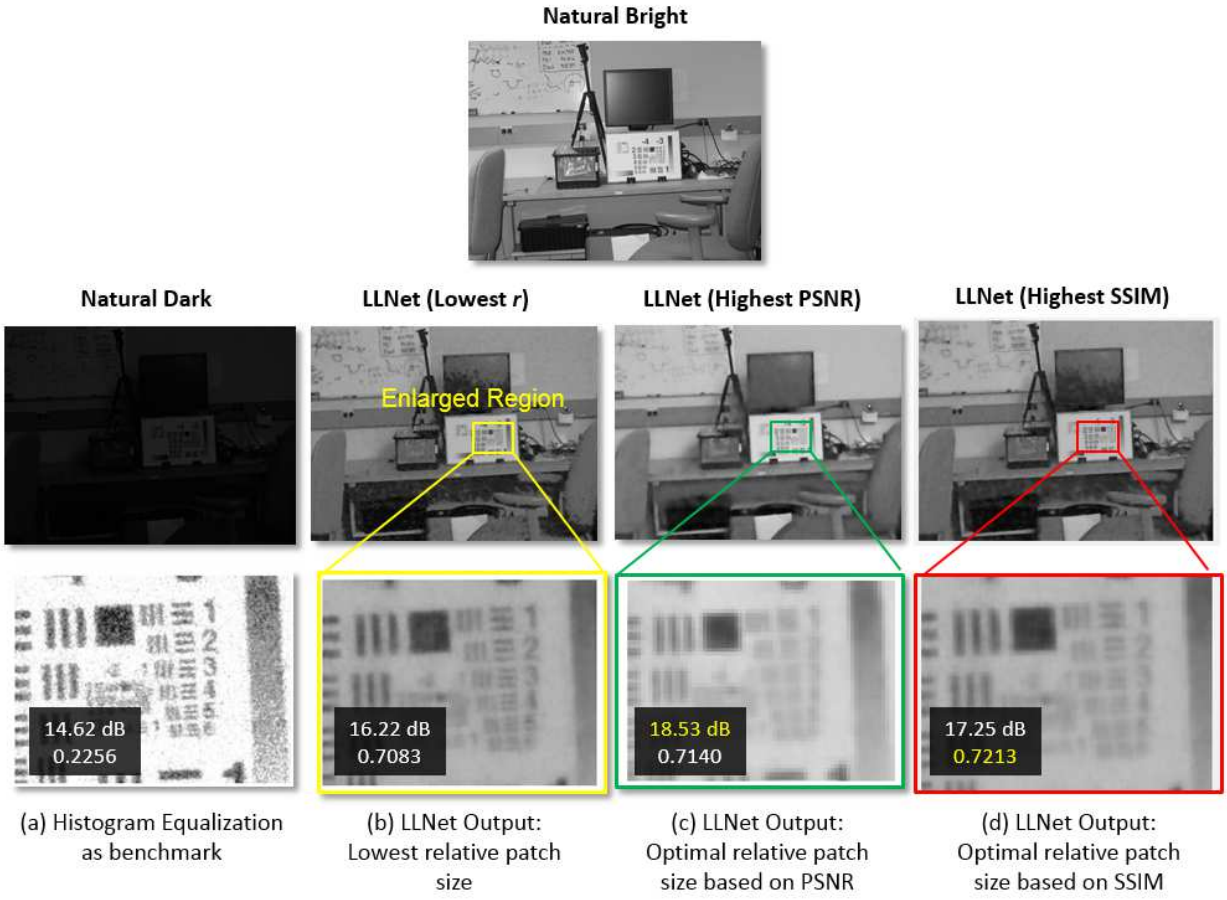


Figure 3.7: Evaluation on US Air Force (USAF) resolution test chart. There exist optimal relative patch sizes that result in the highest PSNR or SSIM after image enhancement (using LLNet). Note that the result enhanced with histogram equalization is shown to highlight the loss in detail of the natural dark image (where the main light is turned off) compared to the natural bright image.

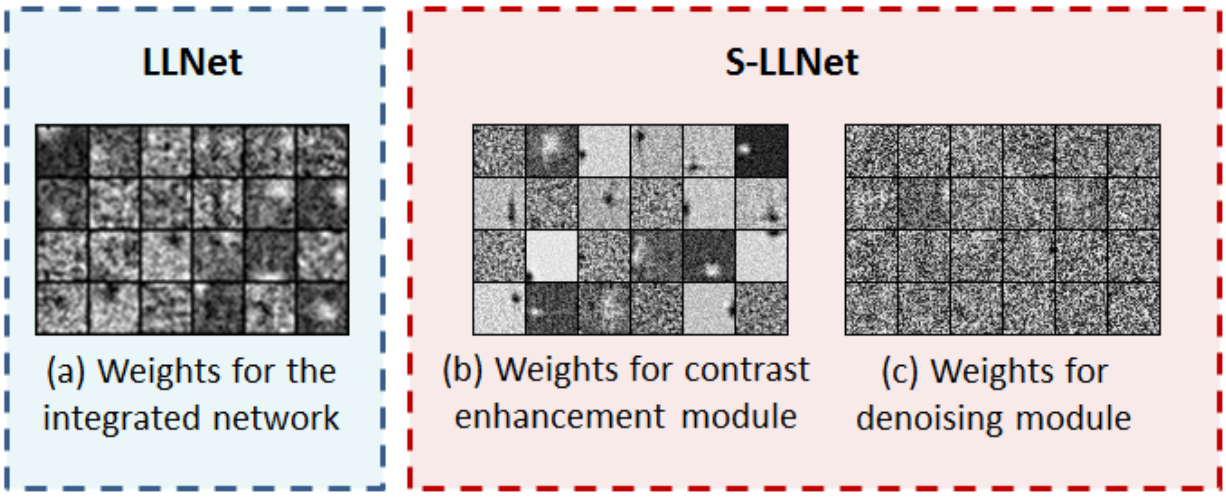


Figure 3.8: Feature detectors can be visualized by plotting the weights connecting the input to the hidden units in the first layer. These weights are selected randomly.

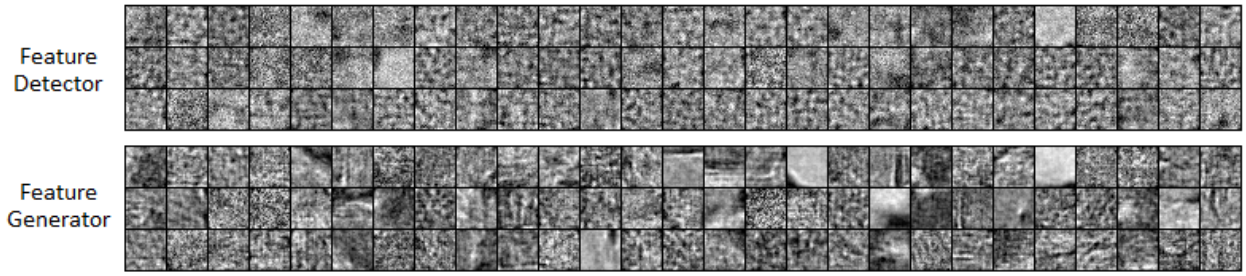


Figure 3.9: Random selection of weights from the first layer (feature detectors) and weights from the output layer (feature generators) for the integrated LLNet model trained with a patch size of 21×21. Patterns in the output weights are similar to patterns in the first hidden layer weights since tied weights are used.

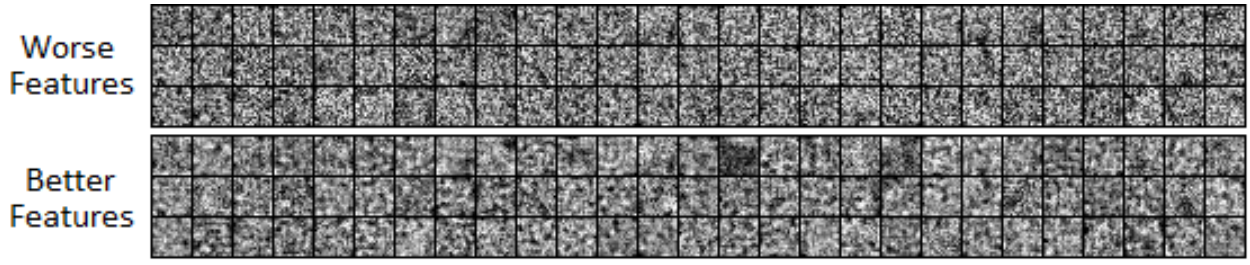


Figure 3.10: Random selection of first layer weights from an integrated LLNet model trained in batches of 1000 and 50, respectively. The superior model (i.e. batch size 50) learns features that appear more distinctive.

CHAPTER 4. FEATURE EXTRACTION FROM SPATIAL DATA – CONVOLUTIONAL SELECTIVE AUTOENCODER (CSAE)

In this chapter, the main part of a proposed convolutional selective autoencoder technique is presented. The ideas behind it are rigorously and methodically formulated to include how it extends to previous neural network principles. Thereafter, some of the achievements of the technique are shown.

4.1 Introduction

Automated labeling of data has been identified as the major challenge of the supervised training techniques (Bengio et al. (2014)). So far, labeling tasks are usually costly or ineffective except when the benefit of crowd-sourcing is leveraged. In a "learning by comparison" setting, a learner properly resolves features that has clear distinctions than those of related features. In that light, we proposed a selectivity function for automatically discriminating the dataset as an image labeling task. The idea is to define a rule to discriminate the signal units of interest from those considered as disturbances in an automated, across-samples basis. The algorithm can serve as a supervised pre-training where the prior knowledge is automatically factored into the architecture. The function is mainly applicable in object recognition and detection tasks, with possibility of adaptation to other classification tasks.

4.2 Selectivity in Deep Networks

Previous computer vision techniques have considered learning from single exemplar (Malisiewicz et al. (2011)) applied to support vector machines for objects detection. However, exploring the ability of deep networks for pixel-wise identification via the recent end-to-end training provided by autoencoders, our selectivity extends image-based exemplar learning to pixel-based or super-pixels

autoencoder architectures. In order to effectively learn the object at the topmost abstraction level, knowledge of the examples are necessary for more efficient training of machines. Each vector in our image is a union of units – those representing the object and those of the undesired objects. Note that the scale of the units could be at the pixel or the super-pixel level depending on the connectivity of the pixels to describe the targets.

4.2.1 Mathematical formulation of selectivity

With this criterion, we reduce the complexity of the objective function of a standard convolutional network to learning the black box that maps different pixel topographies. Given the training data made up of input and target pixel pairs $(X^1, Y^1), \cdots, (X^P, Y^P)$ respectively, which are assumed independent and identically distributed in most of our applications. The dataset is composed of exemplars, (Malisiewicz et al. (2011)) that are: whole images, patches or super-pixels that have the full extent of the object of interest, d. Here, however, the patches mostly have the interest objects, d or unwanted objects, $\widetilde{d}$ such that , $Y^i \subseteq (Y_d^{(s)} \cup Y_{\widetilde{d}}^{(s)})$ and $X^i \subseteq (X_d^{(s)} \cup X_{\widetilde{d}}^{(s)})$, since $\exists$ varieties of d's and $\widetilde{d}$'s that are not captured by the individual examples. The bracketed subscripts (s) represent the super-pixels extract from the patches of examples and the lower cases x and y are used for unit levels images and targets such that $X^{(s)}_{d(or\ \widetilde{d})} = \{x_j : j = d(or\ \widetilde{d})\ \forall\ j\ \in\ Y^i\}$. Note that the only assumption is the knowledge of what Y^is represent, being the training targets. Probabilistically, the desire was to selectively learn $P(y_d)$ that approximates the joint distribution of $P(y_d, y_{\widetilde{d}})$. Furthermore, the expression can be imagined at the pixel level by assuming that the functions are unique to each feature's class. In such case, selectivity encodes the knowledge of undesired units by equating their distribution to a uniform 0 distribution. The goal is equivalent to ensuring that the mutual information between estimate, $\hat{y}_d$ and corresponding target, y_d are maximized. Also, it is desired that the machine minimizes the information correspondence between $x_{\widetilde{d}}$ and corresponding $y_{\widetilde{d}}$. Selectivity however involves more complexity, being a pixel-level discrimination while most deep learning applications have been tailored to discriminate at the super-pixel level (Farabet et al. (2013)). In such probabilistic setting, objective function ought to take the

form $\arg\min_\theta \sum P(y_d)log(P(\hat{y}_d))$, but this only learns the function to identify d. Also, note here that we could have resorted to a distribution around the known pixels for applications having less informative domain knowledge. The activation rule that is implicitly sent to the deep network would be to learn a function, F such that some random combinations of x_d and $x_{\widetilde{d}}$ in any example maps to y_d and the uniform distribution respectively, in a pixel-wise manner. The oscillation that may result is reduced by the rectified linear unit activation of deep networks, which incidentally has similar activations. The selectivity-based function, F is defined as:

$$F(X^i) = \begin{cases} \hat{y}_d, & \text{if } i = d \\ 0, & \text{if } i = \widetilde{d} \end{cases} \tag{4.1}$$

The function is similar to clippers in electrical circuits as it dampens when it transitions to the undesired units. For cases where there is large similarity between the encoded features of x_d and $x_{\widetilde{d}}$ (at the pixel level), the bounds are relaxed by discriminating on the super-pixel level to define the selectivity function as:

$$G(X^i) = \begin{cases} X_d^{(s)}, & \text{if } i = d \\ 0, & \text{if } i = \widetilde{d} \end{cases} \tag{4.2}$$

In the above described cases, it is the function F(or G) for all the training examples that we desire to learn with the convolutional autoencoder network. Clearly, the function serves as a pre-conditioner for a more object-targeted learning by the network. The selectivity function, F(or G) would have some obvious disadvantages at units that are on the boundary of the desired objects; but the architecture used is able to gracefully handle such transitions.

In the test example, the learned functions are applied to the images which consists of both the desired and undesired objects as in the case in Equation 4.2 in order to learn only the desired objects. In this case, encoding selectivity for each individual unit would lead to significant feature-crafting. This is because while in the training sets, the assumption was that each of the pairs contained in the training data has examples that are composed largely of either d-examples or $\widetilde{d}$-examples only, test samples usually contain both the d-examples and $\widetilde{d}$-examples in the same local

neighborhood at the pixel or super-pixel level. This is usually the case with realistic dataset. Then, we can apply the model learned from the selectively discriminated training examples to determine what examples occur at every position of the local neighborhood relatively to its adjacent neighbor given the knowledge of over-complete kernels.

Lastly, one may note that no assumption concerning the additive or multiplicative nature of the disturbances to the original pixel has been made in the formulation. A review of the autoencoder types including training strategies for admitting the selectivity function is provided in the following subsection.

4.2.2 Convolutional selective autoencoder (CSAE)

CSAEs utilize the benefits of the convolutional autoencoder 2.6.4 to learn the objective for performing a specific task. One essence of selectivity is to ensure that parameters in convolutional autoencoders are not identity functions, like the denoising autoencoder Vincent et al. (2008) aims to avoid. For the convolutional autoencoder, the input, X is replaced with the activation from the output layer, say $\hat{Y}$. In this case, the loss function to be minimized is:

$$\theta^* = \arg\min_{\theta} J(\hat{Y}, Y; \theta) = \begin{cases} \arg\min_{\theta} J(\hat{y}_d, y_d), & \text{if } i = d \\ \arg\min_{\theta} J(\hat{y}_{\widetilde{d}}, 0), & \text{if } i = \widetilde{d} \end{cases} \tag{4.3}$$

The composite cost function, $J(\hat{Y}, Y; \theta)$ for this case is a squared-error function Subsection 2.5.1.1 that was minimized for parameter update. The demonstrated cases split the composite cost for each of the pixel classes. Our formulation translates to the assumption that $P(y_d, y_{\widetilde{d}}) \sim P(\hat{y}_d)$ after the training examples have been normalized in the preprocessing part of the deep architecture. The optimization and weight update algorithms are therefore suited to the usual schemes that have been applicable in deeper networks. The robustness in application of the method justifies the hypothesis since we are able to, in most cases, "effectively" identify the desired objects or structures. The networks are usually large due to the redundancy in filter sizes and their depths. Therefore, training is done on Nvidia's Titan Black and/or Titan X GPU with 2880 and 3720

CUDA cores respectively, in a machine that is equipped with 16MB video RAM, using the python-based machine learning frameworks such as Theano, Lasagne and NoLearn (Bergstra et al. (2010b); Thoma (2016)). Lasagne offers a wide variety of control over the layer types, nonlinearity types, objective functions, interfacing with Theano and its supporting function. NoLearn, on the other hand, is a coordinating library for the implementation of the layers in Lasagne which offers model visualization features.

In the following sections, we discuss and presents results of the convolutional selectivity autoencoder network as utilized for a cyber-agricultural pest management problem and an electromechanical combustion problem as a way of validating the effectiveness of the technique.

4.3 An End-to-end Convolutional Selective Autoencoder Approach to *Soybean Cyst Nematode* Eggs Detection

The end-to-end convolutional selective autoencoder approach is adapted for this application. The autoencoder is trained with expert-labeled microscopic images to learn unique features related to the invariant shapes and sizes of SCN eggs, without any hand-crafting. Its deployment resulted in an efficient rare object detection framework which aids in automated high-throughput detection of the SCN eggs. The proposed framework reduces SCN eggs density estimation cost and expedites the overall phenotyping process significantly in addition to its potential to eliminate the overhead of Fuschine dye addition. For the specific problem, we adapted the convolutional selective autoencoder network: for automated phenotyping on the new impactful plant science application involving rare object detection in microscopic image frames cluttered with debris (disturbances) that have great similarities with the objects of interest (typically $\ll 5\%$ SCN eggs among all objects); by a novel selectivity criterion for efficient training of convolutional autoencoder in rare object detection applications that are complicated by high egg to debris similarity; through the adaptation of standard machine learning performance metrics (important for effective communication to the plant science community) for applications involving image based phenotyping.

4.3.1 Motivation

There is growing interest by breeders, farmers and agronomists in developing technological tools to automatically monitor and quantify biotic and abiotic stresses on plants using non-destructive phenotyping methods in a rapid, high throughput manner. Proponents (Li et al. (2014); Mitka and Bart (2015)) have analyzed the untapped potential of such technology with methods that leverage current improvement in resolution of images and sensor data taken from the vulnerable plant or field sections. An application that can benefit from automated learning of features from image frames is the detection and quantification of pests called *soybean cyst nematode*. *Soybean cyst nematodes* (SCNs), *Heterodera glycines*, are unwanted micro-organisms that reduce yields of a major source of food – soybeans. In the United States alone, approximately $1 billion is lost per annum due to cyst nematode infections on soybean plants. Experts have conceived methods of mitigating the losses through phenotyping techniques via SCN eggs density estimation, and then applying the right control measures. Currently, they rely on labor-intensive and time-consuming identification of SCN eggs in soil samples processed onto microscopic frames. However, phenotyping a vast array of fields requires automated high-throughput techniques. From an automation perspective, detection of rarely occurring SCN eggs in a microscopic image frame with a cluttered background of soil debris poses a major technical challenge.

Soybean cyst nematodes have been known (F. W. Nutter et al. (2002)) to compete with the roots of soybean plants for available nutrients causing stuntedness, limiting nodulation of nitrogen fixations, thus leading to huge yield loss of between $30-100\%$ (Grabau (2011); Tylka (2008)). The cysts are formed by dead female worms which, prior to dying, already secreted the eggs and still provide suitable condition for their continuous development. The challenge therefore is to isolate eggs from many other particles in a soil sample. Breeders have sought to detect and subsequently determine the egg densities (Tylka (2008)) from microscopic images, hence they engage workers who manually identify and count the eggs that appear on a microscopic plate after some initial processes that would be described. However, it is tedious, time-consuming and worsened by being significantly error-prone especially when the workers become fatigued. As shown in Figure 4.1, the

objects of interest are largely similar to the soil debris on those frames. Thus, isolating the SCN eggs from other undesired non-eggs particles on microscopic image frames is a complex rare object detection problem that have enormous plant science implications.

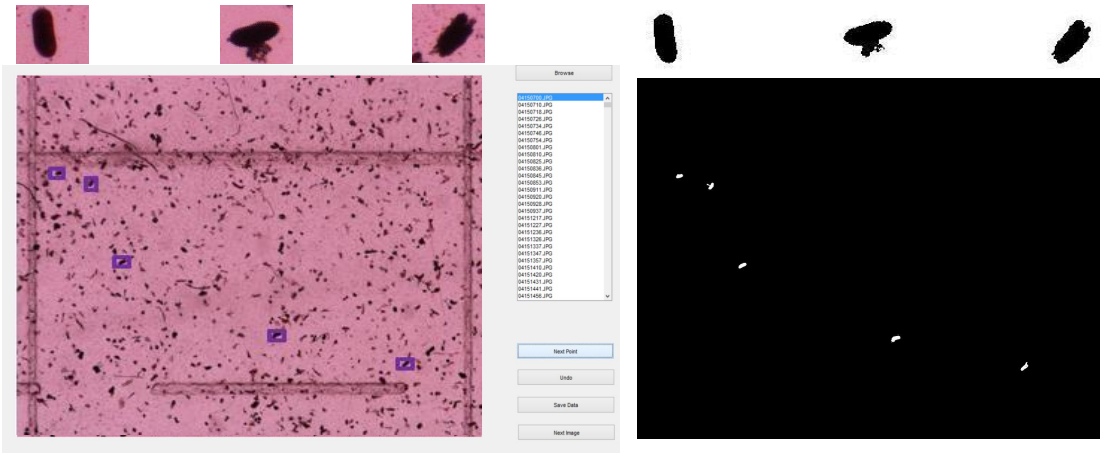


Figure 4.1: Plate a) Shows an example of a frame with eggs in purple boxes and unwanted particles on a *MATLAB*-based GUI for tool-labeling of plant pathologists-identified eggs and Plate b) shows the ground truth labels derived.

In the past, expensive GIS method (F. W. Nutter et al. (2002)) have been used to estimate the density. Various computer vision-based attempts have not been reported due to their failure on the problem. The current image analysis by Syed (2015) included some smart feature engineering based on shape, size and color of the SCN eggs to achieve within 10% of the expert count. Importantly, they also required the overhead cost of adding the Fuschine dye to aid the algorithm identify the features.

4.3.2 Algorithm

A deep convolutional autoencoder improvement of a previously applied stacked denoising autoencoder (Lore et al. (2017)) was trained to suppress undesired parts (non-egg objects) of an image frame while allowing the desired parts (egg objects) for efficient object detection requires an extra selectivity criterion. As shown in Figure 4.3, selectivity is added to supervise autoencoder. The criterion allows the autoencoder to propagate an egg unit if and only if it is fully seen in a

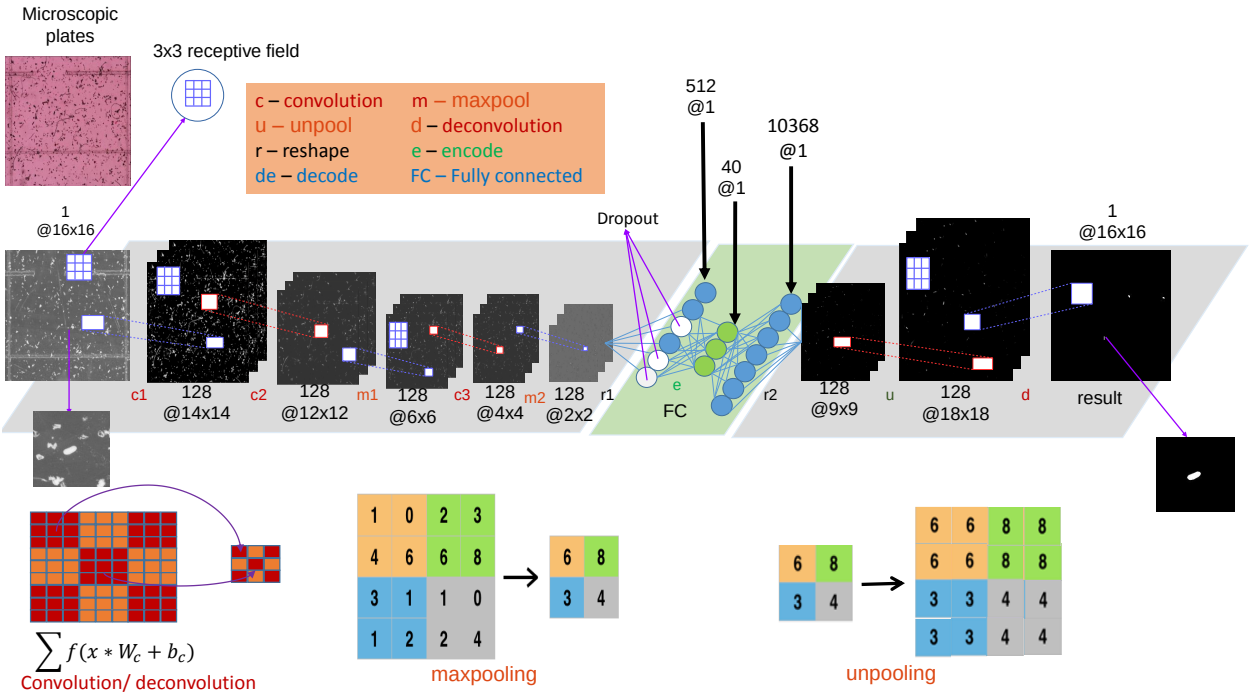


Figure 4.2: Convolutional autoencoder architecture. Letters are used to describe the layer types, c- convolution, m-maxpooling, r – reshape, u – unpooling and d – deconvolution layers respectively and the digits denote the numbered position of that layer among other layers of same alphabet.

patch, and to reject the example either when there is no egg present in a patch or there is an egg that is neither full nor centered. It is included in the algorithm by adding a function that teaches the autoencoder to mask pixels of eggs in the former and propagate pixels in the latter. From a modeling architecture perspective, multi-scale convolutional networks by Farabet et al. (2013) for scene labeling are performed on super-pixels while selectivity requires pixel level. Like most semantic segmentation approaches (Farabet et al. (2013); Pinheiro and Collobert (2014)), selectivity helps to gain more benefit from higher resolution analysis which is required for detection in this close similarity type scenario. The pixel-level classification in turn required an end-to-end type autoencoder where the propagation on pixel levels are projected down to the output layer to solve the task. Also, it is clear that applying multi-scale architectures to this problem would be detrimental to the performance objective of a selectivity autoencoder since partially seen eggs are shown as eggs in this case. Given an $M \times N$ dimensional image frame, P number of patches of

$m \times n$-dimensional were extracted from the image ensuring adequate localization of algorithm on frames. While an original patch is denoted by X^i, the corresponding patch desired to be selective of objects is denoted by Y^i for $i = 1, 2, \cdots, P$. The only pre-processing performed on data pairs $\{(X^1, Y^1), \cdots, (X^P, Y^P)\}$ is normalization to center the pixel intensities around the image statistics. After that, we enhanced egg-similar shapes, sizes and poses only, based on background knowledge with the selectivity criteria through a selectivity function. Those eggs not matching the selectivity criteria are considered to be included as negative examples since such examples would then share many of the qualities of the disturbances. Then, the convolutional selective network architecture described in Subsection 2.6.3 and tailored to this application as shown in Figure 4.2 was applied.

4.3.3 Dataset and implementation

4.3.3.1 Dataset generation

The dataset is typically generated using a 1-inch diameter soil probe to collect soil. Soil was collected during Fall 2015 from random placement of soil probe within several farms in the state of Iowa, where there are varying levels of SCN infestation. Soil samples are mixed together in a bag and washed with water. SCN eggs and cysts are partially separated from the soil in a semiautomatic elutriator Bryd et al. (1976). A motorized rubber stopped Faghihi and Ferris (2000), an example of a physical method is then used for releasing the eggs from the cysts. Purple fuschin dye is thereafter applied to the partially cleaned soil containing the eggs. $1ml$ sample is placed which in each hemocytometer and images of the sample were taken using a camera, or eggs counted under a microscope. About a thousand images were collected using this protocol. These images were then labeled by trained plant-pathologists. The images were labeled by carefully screening each of them, and identifying the location of every SCN egg present in that image. To enable efficient labeling, a *MATLAB*-based app with graphic user interface (GUI) shown in Figure 4.1 was created to simplify identification and marking of the SCN eggs location in the image. The app design included a user-friendly way of selecting images, zoom functions, drawing a rectangle region of interest over the

eggs, saving the location and skipping an image if no SCN eggs were found. The app was deployed on a touch screen enabled device like the Microsoft Surface Pro, allowing the plant-pathologists who detects the eggs physically to just use their fingertips for rapid labeling. The bounding boxes of every SCN egg in $\approx$ 1000 images were extracted and stored as training examples for the network.

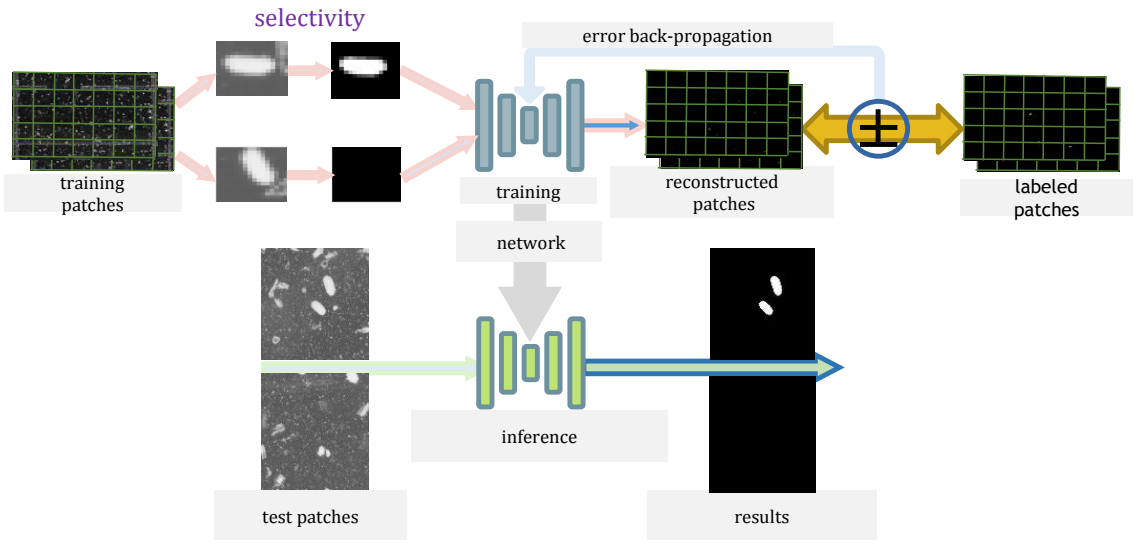


Figure 4.3: Flow diagram of algorithm's implementation.

4.3.3.2 Training and Implementation

Training dataset was divided into the cropped and un-cropped images with image labels. All datasets had patch sizes, $(m \times n) = (16 \times 16)$. The total set available to train the model is shown on Table 4.1 after the data augmentation which includes rotation of each egg between $0 - 180°$ to cover the input space of its variety in orientation as this helps reduce parameters to learn. Training dataset was made up of 80% for training and 20% for validation.

Best learning rate for algorithm was found to be 0.002 at momentum rate of 0.975 and algorithm was trained to 200 epochs with the opportunity for the SGD update depending on the accuracy level. The training was done on GPU Titan Black with 2880 compute universal device architecture (CUDA) cores, 6GB memory, in the theano (Bergstra et al. (2010b)), lasagne and nolearn wrappers

Table 4.1: Training set breakdown: cropped – S, translated – T, rotated – R and labeled – L.

SET TYPE	ORIGINAL DIMENSION	FINAL DIMENSION
S,T,R & L	45432×10 ROTATIONS	$454320 \times 16 \times 16$
S, R & L	2524×10 ROTATIONS	$25240 \times 16 \times 16$
L ONLY	$634 \times 480 \times 640$	$760800 \times 16 \times 16$
TOTAL		$1240360 \times 16 \times 16$

Thoma (2016) of python based on improvements described in section 4.3.2. Lasagne had the layer details, nonlinearity types, objective function, theano extension and many more built into it. Nolearn on the other hand was a coordinating library for the implementation of the layers in lasagne including the visualization aspects. In the training section, a $(c \times c) = (3 \times 3)$ filter size and a non-overlapping $(p \times p) = (2 \times 2)$ were found to be experimentally less costly to produce the results. Algorithm training was done in batches of 128 patches which was found to be suitable. The trained model had overall 743209 learned parameters.

4.3.3.3 Testing set

Test sets available had dimensions, $P = U \times V$, at patch sizes, $m \times n$ and strides of $s_h \times s_w$, where U and V are the final vertical and horizontal number of patches respectively. The dimensions are expressed as: $U = \frac{(M-m+s_h)}{s_h}$ and $V = \frac{(N-n+s_w)}{s_w}$ where $M \times N$ represent the size of each image frame. The flow chart in Figure 4.3 shows the schematics of the overall implementation of the tool-chain including patching, cnn training and the post-processing steps.

4.3.4 Results and discussions

In this section, performance of the convolutional selective auto-encoder is presented and analyzed with respect to the SCN egg detection problem. In order to reduce the false alarms that occur when non-egg particles have high degree of similarity with the eggs, a threshold value close to the maximum is applied to all the results. Our analysis is divided into two main parts: detection effectiveness (from an algorithmic perspective) and computation time and accuracy (from an application requirement perspective). Before discussing the algorithm's detection effectiveness,

a justification of the pipeline's ability to reproduce the patch blocking training is shown in the Figure 4.4. Note that this tracks back to the selectivity criterion described in Section 4.2 where pose centering was included in the training set in order to reject disturbance-like SCN eggs unless such egg is fully centered by a neighboring scanning filter.

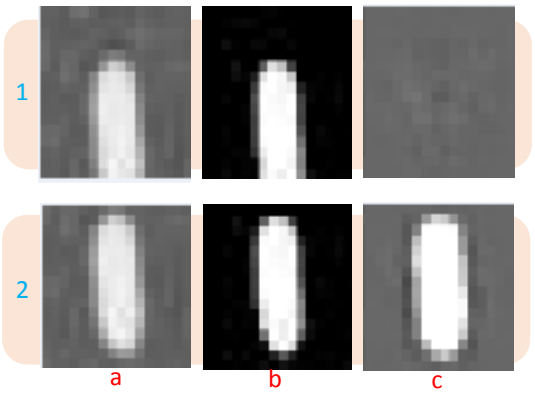


Figure 4.4: Plate 1(a) uncentered patch, 1(b) ground truth, 1(c) algorithm output demonstrating the rejection capability for pose uncentered SCN eggs; 2(a) centered patch, 2(b) ground truth and 2(c) algorithm output demonstrating selectivity of our algorithm on the same translated SCN egg.

4.3.4.1 Detection effectiveness

Egg detection results obtained from the cnn-based tool-chain for some of the testing sets are reported in Figure 4.5. Results shown on plates of Figure 4.5a and 4.5b, where the algorithm captures the eggs (only) in all but one case shows its effectiveness in suppressing the neighboring non-egg particles. The properties of shape, pose and illumination were evaluated by the algorithm to differentiate the non-egg particles from the eggs substances. This would especially be true in the local neighborhood of the eggs where any influence of highly similar clutters would easily have worsened the result. A common thresholding gray-scale value of 180 was used for two observed reasons; 1). the set of egg shapes and sizes are in-exhaustive, such that the training examples may not contain all varieties of SCN egg sizes and types, and 2). the optimization does not

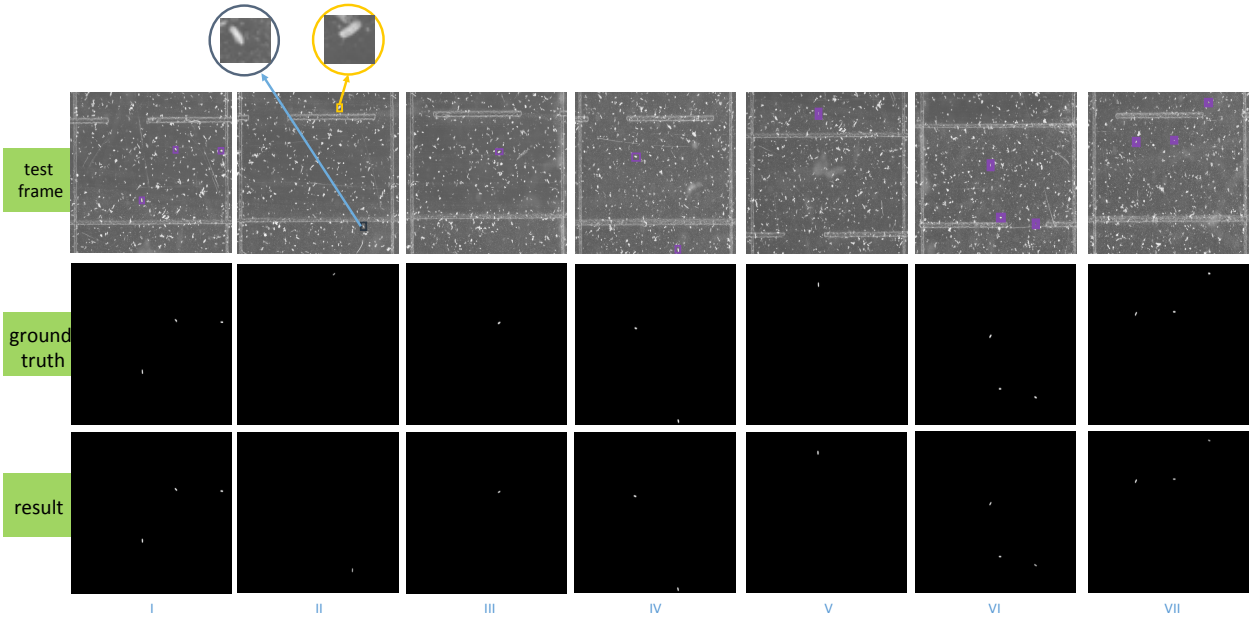


Figure 4.5: Detection results with purple boxes indicating correctly labeled eggs, yellow and blue boxes show missed detections and false alarms respectively.

completely reduce the training error to zero. With the threshold set depending on level of frame clutter, it is encouraging that the algorithm is able to visually signify high confidence level for egg particles and low confidence for disturbances that are really similar to eggs. Farmers may thus make their own judgments on some of the probable defects in locating the eggs to remove the bias in detection. We have highlighted the only miss and false alarm on frame II to show how similar a disturbance usually is to the eggs. The effects of other smaller clutters have been eliminated by the addition of selectivity criterion during training. Note that we however have disregarded that and reported the result without any individual frame thresholding. A possible shortcoming of our proposed framework could be the non-detection of boundary situated eggs. However, zero-padding at boundaries is an option included to extend each test frame beyond its boundaries. This ensures that those eggs with edges directly on the boundary of the frame are centered. The result of such padding is shown in Figure 4.6, but this has an extra cost of having more patches which may not always be beneficial for the desired inferencing speed. Figure 4.6 shows an instance of a boundary situated egg. A low activation of the plate (c) causes a missed detection of the boundary situated egg

due to lack of zero-padding at the boundary, whereas end correction enables a successful detection as shown in plate (d).

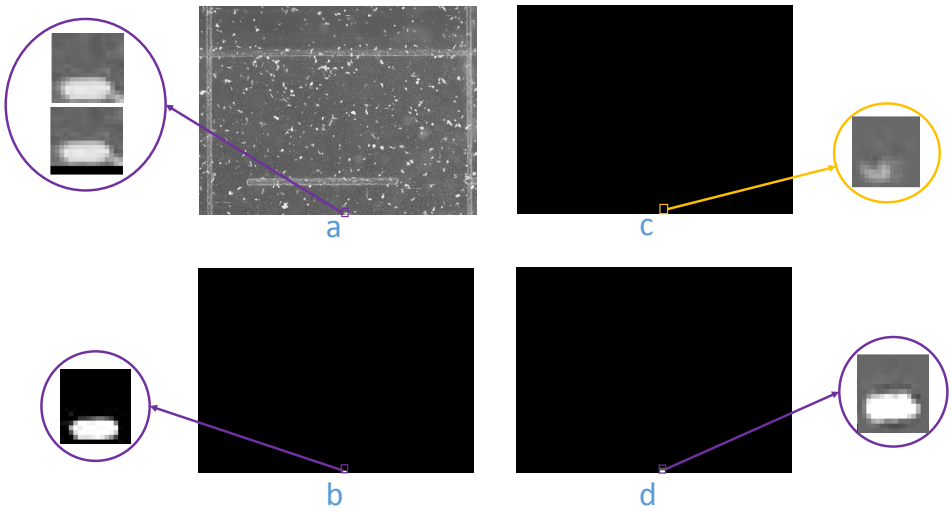


Figure 4.6: Boundary situated object (egg) scenario: (a) test frame, (b) ground truth with egg shown in a purple box, (c) missed detection (shown in yellow box) due to lack of boundary padding and (d) successful detection (shown in purple box) as a result of end correction.

4.3.4.2 Computation time and accuracy

While training the cnn model on the data sets takes several hours, inferencing on new frames is faster as required for high throughput operation. Testing patches were created with an adequate strides of ($s_h = 2$, $s_w = 2$) particularly required for the level of clutter. A ($s_h = 4$, $s_w = 4$) for instance would mean that there is higher chance of having partially enclosed eggs. With smaller step sizes however accuracy increases at a cost of increased complexity and consequent time increases. Formally, the computation complexity for prediction due to the patching step can be described as:

$$\mathcal{O}(\frac{(M - m + s_w) \times (N - n + s_w)}{s_h \times s_w}) \approx \mathcal{O}(M \times N) \tag{4.4}$$

Table 4.2 shows the detection times required for an image frame with different patch strides. A well-trained expert plant-pathologist is estimated to take in the order of 5 minutes to examine a frame while the worst detection time of the algorithm for strides ($s_h = 1$, $s_w = 1$) shown would be

for extremely cluttered image frame. The gains recorded are usually over large number of images covering the infected land area. In most test cases, the (2×2) stride provides a good trade-off

Table 4.2: Detection time required for an image frame with different patch sizes.

STRIDE	P - #OF PATCHES/FRAME	DETECTION TIME(SEC)
1×1	290625	236
2×2	72929	55
4×4	18369	14
8×8	4661	5
16×16	1200	2

between computation time and accuracy. Typical performance metrics used for object detection tasks such as the accuracy and the confusion matrix may be inadequate due to the overwhelming presence of non-egg particles. Breeders usually consider certain metrics to be more significant than others. Therefore, we adapt some of the standard classification performance metrics in order to effectively communicate results to them. The metrics used are: average detection accuracy, ADA

$$= \frac{\text{average detection/frame}}{\text{average actual eggs/frame}}$$

average alarm-to-egg ratio, AAER

$$= \frac{\text{average false alarms/frame}}{\text{average detection/frame}}$$

average miss-to-egg ratio, AMER

$$= \frac{\text{average misses/frame}}{\text{average detection/frame}}$$

average precision, AP

$$= \frac{\text{average detection/frame}}{\text{average detection/frame+ average false alarm/frame}}$$

The result for all test frames is computed and shown in Table 4.3 based on over 100 unseen frames. Note that the selected threshold gives a higher preference to the detection accuracy compared to lowering false alarms. The rationale is that with a low missed detection probability, the resulting frames can be quickly examined by the experts to reject the false alarms (which is drastically low in number compared to the non-egg objects in the original frames) and still have a reliable

Table 4.3: Performance metrics for SCN egg detection problem.

Standard Metric	Adapted Metric	%	Significance
Recall	ADA	≈ 100	This is an accuracy measure as the breeders are interested in knowing how close to the true number of eggs an algorithm can detect. The level of accuracy affects their timely and correct action in order to reduce the overall yield loss in soybean farms. Note, ADA is approximately 100% as the amount of false alarm and missed detection compensate for each other.
False positive ratio	AAER	≈ 5	It indicates the percentage of debris that the algorithm misclassifies as eggs. Breeders are interested in reducing false alarms such that their control actions do not become overpowering and have unnecessary side-effects.
False negative ratio	AMER	≈ 5	It indicates the percentage of eggs that the algorithm misses to detect. Breeders are interested in reducing such missed detections such that their control actions do not become insufficient and result in yield loss,
Precision	AP	≈ 96	It indicates the accuracy level from the algorithmic design perspective, the goal is to increase the detection accuracy while reducing the amount of false alarms.

count of eggs. One of the assumptions of this selective autoencoder framework is that the patch size must be at least same or larger compared to the size of the largest SCN egg to be detected. Subsequently, the algorithm has been tested to perform similarly on several more images than could be reported because of the non-availability of ground truth.

In summary, an end-to-end convolutional selective autoencoder approach was leveraged for a complex rare object detection problem in the domain of automated phenotyping. The embedded selectivity criterion enabled the autoencoder performance on the task. Also, hyper-parameters for the convolutional network are meticulously explored for the critical plant science problem of automated phenotyping of a particular soybean biotic stress on microscopic images containing soil samples. The machine learning pipeline uses expert-labeled training examples (with the possibility of human errors) and can serve as a decision support tool that has potential of saving enormous amount of time for agricultural/plant scientists in characterizing a significant disease affecting soybean yield in the United States. Such automated, high-throughput phenotyping of the SCN eggs can also assist farmers to determine the soybean varieties or cultivars that are resistant to the stresses induced by the worm. These resistant varieties can then be reproduced for planting with high assurance of minimal yield losses to the infections from SCN. From a machine learning perspective, a typical image frame in this application mostly contains objects that are extremely similar in all properties to the objects of interest (SCN eggs). Features hand-crafting constitutes a very difficult proposition in those scenarios, hence deep learning technique was the appropriate choice. Some of the future plans at the time of research submissions were to: (i) improve pre-processing the object patches via learning optimal transformation for a more efficient detection;(ii) adapt toolchain for general rare object detection and estimation; (iii) develop an easy interface such as a smart phone app for plant scientists/farmers to estimate the densities of eggs in an online fashion.

4.4 Prognostics of Combustion Instabilities from Hi-speed Flame Video using A Deep Convolutional Selective Autoencoder

In this research, we developed a deep convolutional selective autoencoder-based anomaly (early) detection framework for the crucial physical process of combustion for an improved understanding of the underlying complex physics. This was achieved by capturing the rich information in hi-speed flame video for instability prognostics. In this context, the autoencoder is trained to selectively mask stable flame and allow unstable flame image frames. Performance comparison is done with a well-known image processing tool, conditional random field that is trained to be selective as well. For comparison purpose, an information-theoretic threshold value is derived. The proposed framework is validated on a set of real data collected from a laboratory scale combustor over varied operating conditions. The network effectively detects subtle instability features as a combustion process makes transition from stable to unstable region.

The contributions of the research are summarized as follows.

- A convolutional selective autoencoder framework based on emerging deep learning techniques is proposed for a significant PHM application – early detection of combustion instability;
- The method avoids extensive expert-guided feature handcrafting Farabet et al. (2013) while addressing a complex physical phenomenon like combustion to discover coherent structures in flames images;
- The proposed framework is able to learn from high dimensional data sets (e.g., high speed video) of most applications and provides a platform for determining the degree of relationship between the states of two temporally close observations;
- A metric to desired level of granularity is constructed to track the onset of combustion instability and detect pre-transition phenomena such as 'intermittence'. Intermittence is a temporary (of the order of millisecond, equivalent in this case to few video frames) blast of instability characterized by small and partially observable coherent structure;

- Extensive validation and comparison using CRF technique are provided based on laboratory-scale combustion data collected under various realistic operating conditions.

4.4.1 Motivation

Combustion instability was found to be a significant anomaly characterized by high-amplitude flame oscillations at discrete frequencies that reduces the efficiency and longevity of aircraft gas-turbine engines. Full-blown instability can be differentiated from stable combustion via video analysis with high confidence because unstable combustion flames show distinct coherent structures similar to 'mushroom' shapes. But it is extremely difficult to detect an onset of instability early due to fast spatiotemporal transience in the video data. Therefore, the instability detection problem boils down to an implicit soft labeling problem where we train a deep model using hi-speed flame videos with explicit labels of stable and unstable flames such that it recognizes the onset of instability early as the combustion process makes transition from a stable to unstable region.

Combustion instability reduces the efficiency and longevity of aircraft gas-turbine engines. It is considered a significant anomaly characterized by high-amplitude flame oscillations at discrete frequencies. These frequencies typically represent the natural acoustic modes of the combustor. Combustion instability arises from a positive coupling between the heat release rate oscillations and the pressure oscillations. Coherent structures are fluid mechanical structures associated with coherent phase of vorticity (Hussain (1983)). The generation mechanisms of the structures vary system wise, causing large scale velocity oscillations and overall flame shape oscillations by curling and stretching. These structures can be caused to shed–or be generated–at the duct acoustic modes when the forcing (pressure) amplitudes are high. There is a lot of recent research interest on detection and correlation of these coherent structures to heat release rate and unsteady pressure. The popular methods resorted for detection of coherent structures are proper orthogonal decomposition (POD) (Berkooz et al. (1993)) (similar to principal component analysis (Bishop (2006)) and dynamic mode decomposition (DMD) (Schmid (2010)), which use tools from spectral theory to derive spatial coherent structure modes.

The thermo-acoustic instabilities arising in combustion processes cause significant deterioration and safety issues in various human-engineered systems such as land and air based gas turbine engines. The phenomenon is described as self-sustaining and having large amplitude pressure oscillations with varying spatial scales of periodic coherent vortex shedding. Early detection and close monitoring of combustion instability are the keys to extending the remaining useful life (RUL) of any gas turbine engine. However, such impending instability to a stable combustion is extremely difficult to detect only from pressure data due to its sudden (bifurcation-type) nature. Tool-chains that are able to detect early instability occurrence have transformative impacts on the safety and performance of modern engines.

Previous works have modeled it with power-law distribution, small world-like nature (Okuno et al. (2015)), nonlinear dynamics such as bifurcation and limit cycle properties (Nair et al. (2014)) or using physical properties such as Rayleigh's (or Pseudo-Rayleigh (Okuno et al. (2015)) index computed at varying frequency and operating condition (Poinsot et al. (1987)) and phase between quantities, which are neither generalizable in all cases nor easily adaptable for automated identification of the structures. The chaos at the flame front may be responsible for this observation. As the chaos becomes fully developed in the unstable region, the evolution becomes almost unpredictable via models. Data-driven techniques are more appealing for analyzing the instability onset. Statistical methods have been developed to indicate the onset of transitioning of combustors from stable regimes. Such statistical approaches have explored correlation properties (covariance matrices) or higher order statistics such as skewness and kurtosis (Brock and Carpenter (2006); Bergland and Gentz (2002)) for chaotic interactions. The approaches are however reported (Fox and Whiteside (1987)) to be suitable for the analysis of non-fully developed structures. A possible reason is the assumption that the best explanatory features can be captured by a single space (or scale) of the data using a known statistical order. The properties of fully-developed structures (chaos, bifurcations and limit cycles) are completely time varying (as well scale varying). Therefore, the statistical model required to handle such cases have to be robust to handle such chaos in the coherent structures. Also, the oscillation pathways involved in the thermo-acoustic oscillations are

found (Nair et al. (2014)) to transition via chaos from stochastic to periodic with varying mixture equivalent ratios. Spectral methods have also utilized signal decomposition into mode spectra and frequency resolution (Richecoeu et al. (2012)).

It is still not common to apply the cutting edge improvements of deep learning towards developing advanced *Prognostics and Health Monitoring* (PHM) algorithm for typical engineering applications. In the research, we proposed a novel selective autoencoder approach within a deep convolutional architecture to analyze hi-speed flame videos for early detection of combustion instability in a gas turbine engine. Whereas traditional PHM algorithms mainly use time series data (e.g., pressure and temperature etc.). For this purpose, the proposed approach attempts to advance PHM via capturing the rich information of hi-frequency video. The approach performs implicit labeling in order to derive soft labels from extreme classes that are explicitly labeled as either positive or negative examples. This particular property is significant for tracking continuous temporal phenomenon such as the transition from combustion stability to instability, where labels of extreme states (stable or unstable) are available but intermediate state labels are not. Explicit labels are utilized to selectively mask critical features while allowing other features to remain. Figure 4.7 shows gray-scale images describing typical gradual development of instability at the stated parameters in the swirl-stabilized combustor used for the experiment.

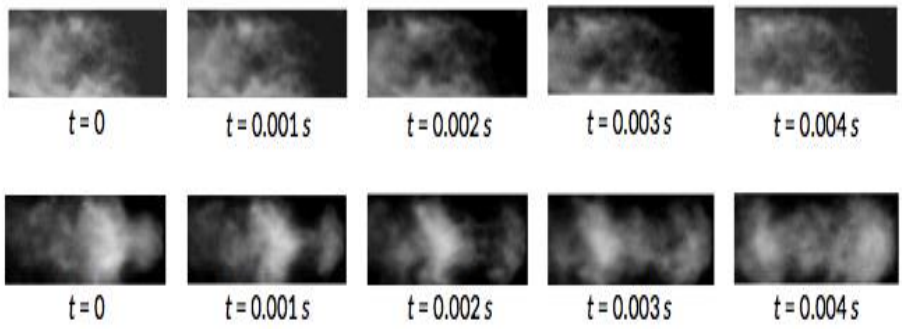


Figure 4.7: Gray-scale images of gradual time-varying development of instability structure at two different parameter values.

As an application of the network for detecting the level of instability in an assumed stable flame frame, image labeling was explored. Labeling (e.g., structured and implicit) can be considered a multi-class classification problem (Erdogan (2010)). For example, three-stage Hidden Markov Models (HMM) were used for handling speech recognition (Rabiner (1989)) problems, parts of speech tagging (Meyer (2012)) and sequence labeling because they derive the relationships from observations-to-state and state-to-state in dynamic systems. Maximum Entropy Markov Model (MEMM), a discriminative modification of HMM, was introduced to overcome the latter's recall and precision problems especially in labeling texts. In those models, conditional probability of the desired labels are learned directly based on the uncertainty maximization idea. Applications of MEMM for natural language processing can be found in (Berger et al. (1996)).

Semi-supervised training for classification takes advantage of the labels at the final layers. A variant of structured labeling by (Kulesza et al. (2014)) called implicit labeling is used to derive soft labels from extreme classes that are explicitly labeled as either positive or negative examples. Explicit labels usually can be utilized to selectively mask one feature, especially that one is not interested in while parsing the class of interest. However, explicit labels on its own can only serve as a classifier for intrinsic classes in the test sets learned from the training set.

Figure 4.8: Illustration of implicit method of generating soft labels.

Implicit labeling considered are similar to the sequence labeling (Erdogan (2010)) with an extra constraint of utilizing prior knowledge provided only by explicit label. It is then fused with convolutional autoencoder architecture algorithm described in Section 2.6.3 to determine the intermediate or transition phases – a mixed breed of a dog and a wolf for instance – and more

importantly to what degree is the animal a dog or a wolf. Thus, it attempts to derive soft labels from expert-informed, hard-mined labels as illustrated in Figure 4.8 with a composite architecture.

Due to "label bias" defects of MEMM, a Conditional Random Field (CRF), which is a joint Markov Random Field (MRF) by Lafferty et al. (2001) of the states, conditioned on the whole observations is later explored. It enabled considering the global labels of the observation as against localization of labels of MEMM (Erdogan (2010)). However, labeling in this case is made computationally complex by the relaxation of statistical independence assumption of the observations which most of the models assume.

Recurrent Neural Networks (RNNs) have been utilized for sequence labeling problems due to its cyclic connections of neurons (Graves (2014)) as well as its temporal modeling ability. Although earlier construction of RNNs is known to have short ranged memory issues and a restrictive unidirectional information context access, formulation of a bidirectional Long Short Term Memory (LSTM) (Graves and Schmidhuber (2005)) resolved such issues. However, this construction adds to the complexity of the model significantly as typically two RNNs get connected through the same output layer.

From the application standpoint, early detection of instability in the combustion chambers of dynamic systems aids anticipative actions for reducing its consequent effects. Visualizing the features that characterizes the intermediate frames of its spectrum is an important approach to unravel the processes that precede instability. The authors in (Sarkar et al. (2015c)) introduced Deep Belief Networks (DBN) as a viable technique to achieve the aim with a view to exploring other machine learning techniques for confirmation. They improved on that by applying a modular neural-symbolic approach (Sarkar et al. (2015b)) in another publication.

Conceptually, this is similar to cognitive psychologists' description of human reasoning in object classification (Tenenbaum et al. (2011)). An example is to consider how a child is taught on intrinsic classes. A similar problem is how to detect a cross breed of dog and wolf and how close the animal is to either of the classes. From an application standpoint, an early detection of engine's combustion instability may be useful for computing the instantaneous values of the remaining useful life, but

the computation is partial since other physical factors of engine use are also important. Therefore, remaining useful life (RUL) computation is beyond the scope of the present problem.

4.4.2 Algorithm

In this section, the algorithms for sequence labeling are described. We provide a little more details of the convolutional autoencoder and its interface with the selectivity criterion. Subsequently, a brief background on the conditional random field (CRF) algorithm is provided. Then, we discuss the information theoretic metrics that facilitate image dimensionality reduction, and the basis for our threshold computation. The end-to-end convolutional selective autoencoder that was tailor-made for the application is shown in Figure 4.9. It was adapted for the current application.

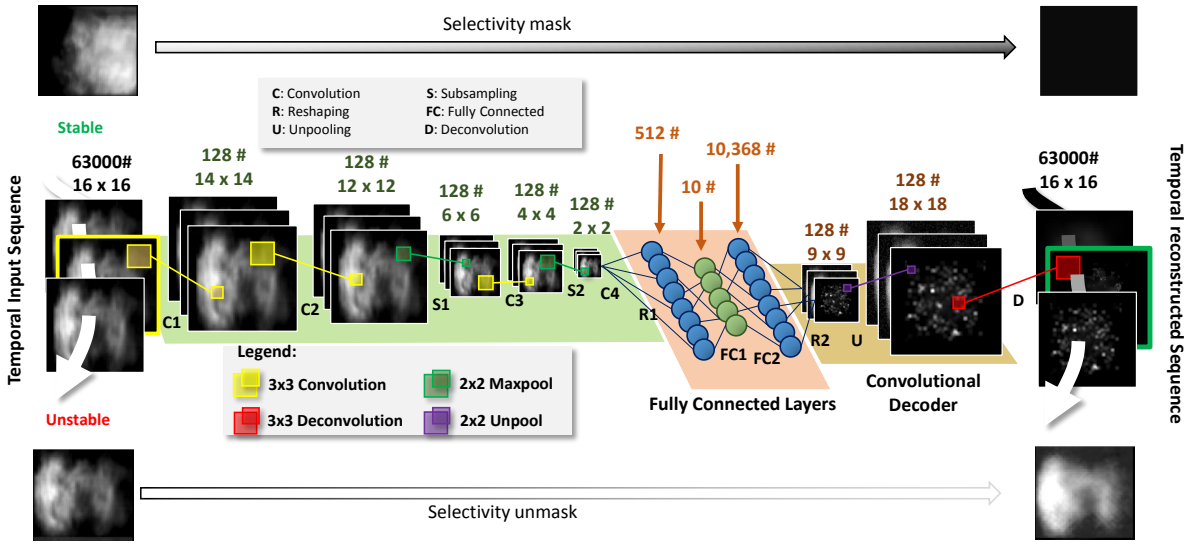


Figure 4.9: Structure of the convolutional autoencoder with selectivity masks. The encoder portion extracts meaningful features from convolution and sub-sampling operations, while the decoder portion reconstructs the output into the original dimensions through deconvolution and upsampling. Best viewed on screen, in color.

Given an $M \times N$ dimensional image frames and corresponding ground truth labels (one of the two classes), explicit labels are generated by selectively masking frames with the undesired class with black pixels. Hence, N pairs of input-output pairs $\{(X^i, Y^i)\}$ for $i = 1, 2, \cdots, N$ are generated where X represents the original images, Y are the masked frames that are considered explicitly as

ground truth. Pre-processing was done by normalizing the pixel intensities in images to have zero mean and unity standard deviation.

4.4.2.1 Conditional random field (CRF)

CRF is another class of well-studied (Domke (2013)) and formulated models for labeling problems. CRF is an improvement on the Markov Random Field, MRF where one is interested in determining the conditional probabilities of newer observation such as our test data given the knowledge of previous ones such as the explicit labels. The benefits of CRF is its improved learning over the likelihood estimation by including inference approximation. The algorithms have been shown (Barbu (2009)) to perform well on complex image problems such as image denoising task as well as being robust to model misspecification. Therefore, we also incorporated selectivity condition into the CRF in a similar way to that of CAE.

4.4.2.2 Instability metric

Similar to that presented in (Liu et al. (2016)), a metric based on the Kullback-Liebler (KL) divergence (Kullback and Liebler (1951)) is chosen to measure the distance of the results from the image frames in each transition protocol from the expected result of a stable flame frame. This yields a KL distance, z for each image frame, $\mathbf{I} \in \mathcal{I}$, where $\mathcal{I}$ represents the set of input images frames. It can be expressed mathematically as,

$$z(\mathbf{I}) = \sum_{i \in \mathbf{I}} \lim_{\mathbf{T}(i) \to 0^+} \mathbf{I}(i) log \frac{\mathbf{I}(i)}{\mathbf{T}(i)} \tag{4.5}$$

where i represents each pixel in the image frame and $\mathbf{T}$ represents the training label/target image. The limit helps to drive a particular flame image pixel values to zero in the stable combustion region. This physically corresponds to taking the distance of each image from the reference of the stable flame. The present metric has the advantage of using a common reference for all the test transition protocols rather than being specific to a particular image frame within one test protocol (Akintayo et al. (2016b)).

4.4.3 Dataset and implementation

In this section, attempts at solving the problem by describing the dataset are motivated, the experimental setup for gathering the data and how it is collected. We also describe the implementation of the two competing algorithms by explaining the choices that were made and stating the important selected parameters for such choices. Finally, the threshold values for analyzing the results are determined.

4.4.3.1 Dataset collection and experimental setup

To collect training data for learning coherent structures, thermo-acoustic instability was induced in a laboratory-scale combustor with a 30 mm swirler (60 degree vane angles with geometric swirl number of 1.28). Figure 4.10(a) shows the setup and a detailed description can be found in (Sarkar et al. (2015c)). In the combustor, 4 different instability conditions are induced: 3 seconds of hi-speed videos (i.e., 9000 frames) were captured at 45 lpm (liters per minute) FFR (fuel flow rate) and 900 lpm AFR (air flow rate), and at 28 lpm FFR and 600 lpm AFR for both levels of premixing. Figure 4.10 (b) presents sequences of images of dimension 100 $\times$ 237 pixels for unstable ($AFR = 900lpm$, $FFR = 45lpm$ and full premixing) state. The flame inlet is on the right side of each image and the flame flows downstream to the left. As the combustion is unstable, Figure 4.10 (b) shows formation of mushroom-shaped vortex (coherent structure) at $t = 0s$ to $t = 0.001s$ and the shedding of that towards downstream from $t = 0.002s$ to $t = 0.004s$. For testing the proposed architecture, 5 transition videos of 7 seconds length were collected where stable combustion progressively becomes unstable via 'intermittence' phenomenon (fast switching between stability and instability as a precursor to persistent instability) by reducing FFR or increasing AFR. The transition conditions are as follows (all units are lpm): (i) AFR = 500 and FFR = 40 to 28, (ii) AFR = 500 and FFR = 40 to 30, (iii) FFR = 40 and AFR = 500 to 600, (iv) AFR = 600 and FFR = 50 to 35, (v) FFR = 50 and AFR = 700 to 800. For clarity, these data sets are named as 500_{40to38}, 500_{40to30}, $40_{500to600}$, 600_{50to35}, and $50_{700to800}$ respectively for analysis in the subsequent sections of this chapter.

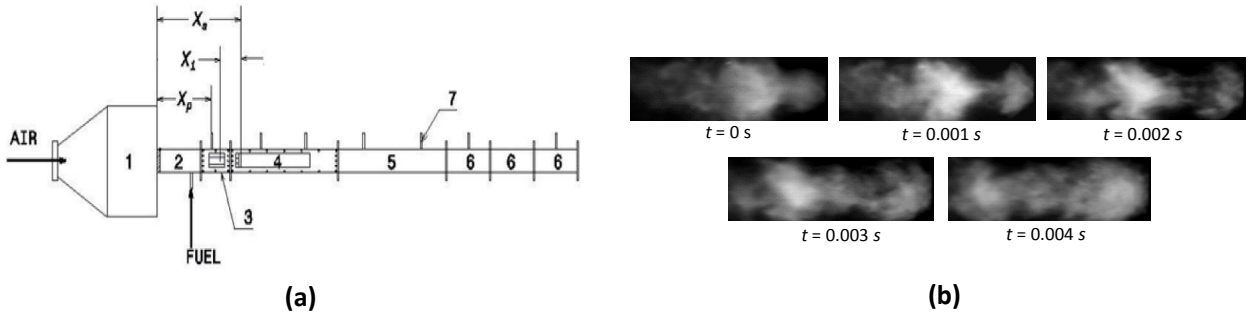


Figure 4.10: a) Schematics of the experimental apparatus. 1 - settling chamber, 2 - inlet duct, 3 - inlet optical access module (IOAM), 4 - test section, 5 & 6 - big and small extension ducts, 7 - pressure transducers, X_s - swirler location, X_p - transducer port location, X_i - fuel injection location, (b) Visible coherent structure in grayscale images at 900 lpm AFR and full premixing for 45 lpm FFR.

However, Table 4.4 is a summary of the various factors and the characteristics of the flame videos that results from the experiments. Note that the experiments were conducted using a swirl-stabilized combustor, but such experiment could potentially be replicated for observations from a bluff-body stabilized Nair et al. (2014) combustor.

4.4.3.2 Training process

In training the networks, $63,000$ gray-scale frames having dimensions 100×237 are resized to 16×16 for computational simplicity. From Table 4.4, total of $35,000$ frames is labeled stable while the remaining $28,000$ were labeled as unstable. These images were a combination of datasets with different premixing lengths of either 90mm or 120mm and a wide range of air and fuel LPMs for which the combustor is either in a stable or an unstable state. The whole training dataset is divided into two parts: 75% of it is used to train the algorithm, while 25% is held out for validating their results and setting our thresholds.

CAE: The parameters of the convolutional autoencoder used include learning rate of 0.0001 with momentum = 0.975 is found to train the model best in the Nesterov based stochastic gradient descent formulation. The network is trained to 100 epochs in order to conveniently strike a good minima of the validation error. Training, a filter of $c \times c$ pixels ($c = 3$ in the implementation) and

Premixing length (mm)	Fuel flow rate FFR (l/min)	Air flow rate AFR (l/min)	State
90	28	600	Unstable
	45	450	Stable
	45	900	Unstable
	60	600	Stable
120	7.5	50	Stable
	28	600	Unstable
	45	450	Stable
	45	900	Unstable
	60	600	Stable

Table 4.4: Breakdown of the experimental dataset used for training the CSAE and SCRF algorithms showing how different combination of factors result in either stable or unstable states.

a non-overlapping $p \times p$ ($p = 2$) maxpooling were found to be experimentally less costly to produce the results. Algorithm training is done in batches of 128 training examples which is found to be suitable via cross validation.

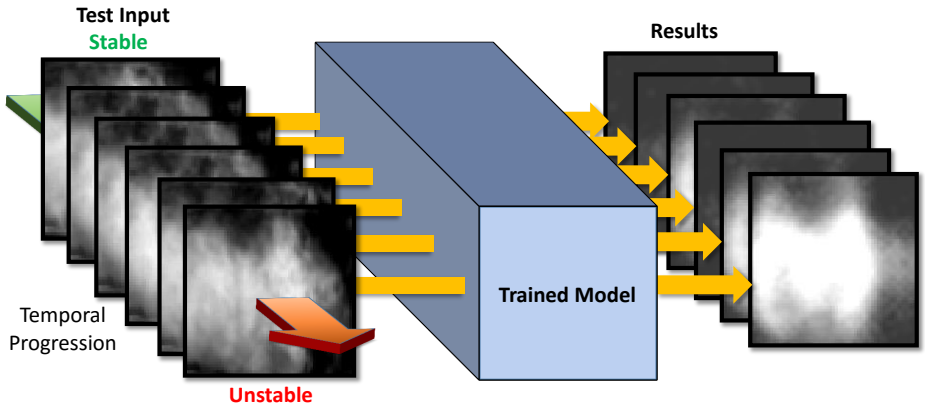


Figure 4.11: Schematics of implementation of trained network on transition test data.

The architecture in Figure 4.9 shows how the layers are interlinked in the training stage which leads to an overall of $416,779$ learned parameters. From this point onwards, CAE model that is trained to be selective is referred to as convolutional selective autoencoder (CSAE).

CRF: In training the linear to linear type conditional random field, the main hyper-parameters are again the loss function which usually is approximated and how the gradient of such objective

function are computed. For the present problem, based on multiple trials for hyper-parameter, we found the loopy variant of the truncated tree re-weighted (TRW) belief propagation a good inference type for the problem. Furthermore, for better performance, we chose a clique type loss because of the benefits over simple univariate type loss. A quasi-netwon method, Broyden-Fletcher-Goldfarb-Shanno (BFGS) was chosen to optimize its error backpropagation. The algorithm is also implemented in batches of 512 to reduce computation time, and in a gradual fashion while the regularization parameter used was 0.0001. The model resulted in 8064 cliques. Subsequently, like the CAE, we refer to a CRF model that is trained to be selective as selective conditional random field (SCRF).

4.4.4 Threshold determination

Given the models learnt from each of the algorithms, CSAE and SCRF individually, with the training sets as illustrated in Figure 4.12, the algorithms are separately validated on the validation set. The validation result for each algorithm is used to determine the value of the instability metric, z at which transition takes place, called transition threshold. This is taken as the upper limit of the 95% confidence interval (CI) for the distribution of z (see Equation 4.5) for stable flame frames. The schematics in Figure 4.12 summarizes how it is implemented for each algorithm. We note that this helps to utilize expert knowledge regarding the stable and unstable regions to determine the start of transition from the stable region.

Note that these are derived by replacing $\mathcal{I}$ in Subsection 4.4.2.2 with the known stable part in the validation results.

4.4.5 Results and Discussions

In this section, results obtained from the algorithms are discussed and analyzed. The subsections are arranged to build up the argument for early detection of unstable region's properties in frames. Such unstable flame properties can be detected even in the transition region enabling early instability detection. Then we discuss how the network explores the space between the stable and

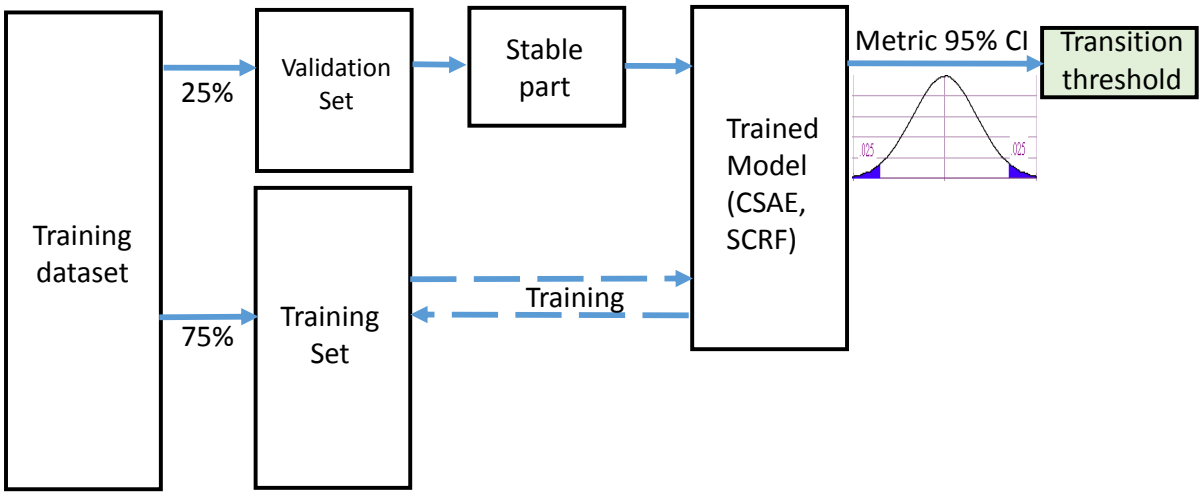


Figure 4.12: Schematics of selection of transition threshold.

unstable regions to get softer labels. Let the stable region be denoted by 'SR' on one end of the spectrum and the unstable region be 'UR' on the other end of the spectrum. Note, training of the algorithm is performed with explicitly available ground truth labels. The ground truth labels are categorized into frames of stable flame types and frames of unstable flame types. As discussed before, units of frames in the stable region are masked with '0', while those in the unstable region are retained during training. Figure 4.13 shows the algorithm's ability to satisfy the training criteria in one stable and one unstable validation frames.

Figure 4.13 shows how CSAE learns to be selective in masking the stable region as trained. Feature maps from the model are shown in Figure 4.14 to highlight the detected features and the reconstructed outputs.

For frames closer to UR in the transition stage, the corresponding feature maps showed more pixels activated mushroom structures that characterize UR. For frames in SR however, information is seen to be rapidly diffusing from the input into the hidden layers. At each layer, joint parameters capture the trade-off between discarded and retained information from the stable and unstable training sets. The fully connected layers serve at least two important purposes, namely: (1) to reduce further the image dimensions towards only rich explanatory features, and (2) ensuring

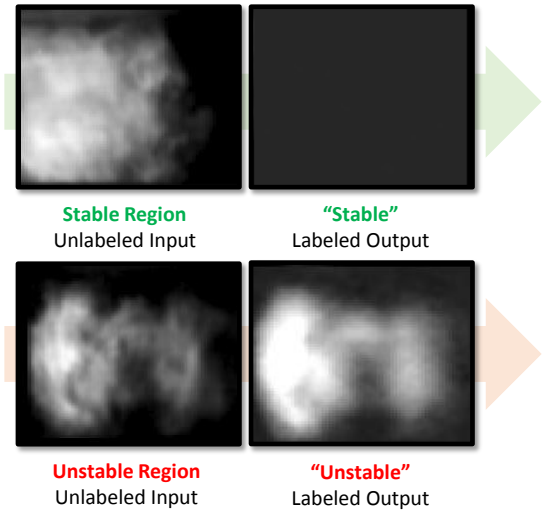


Figure 4.13: Illustration of CSAE's ability to reproduce explicit labels.

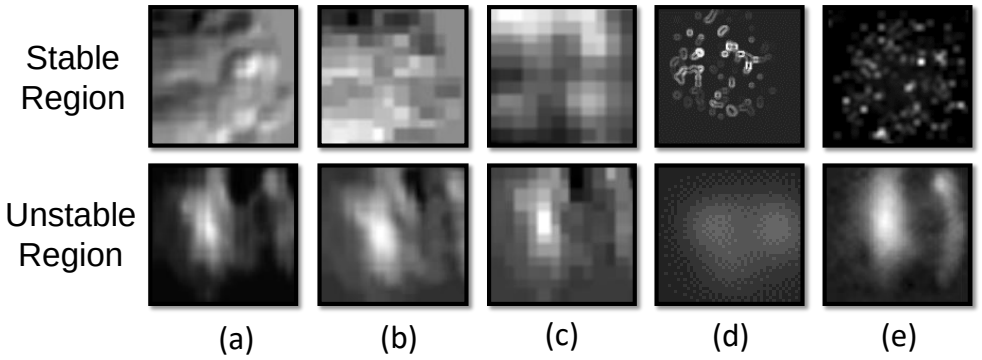


Figure 4.14: Feature maps for (a) the third convolution layer, (b) the second pooling layer, (c) the fourth convolution layer, (d) the unpooling layer and (e) the deconvolution layer.

structural consistency for optimal layer-wise features by reshaping the output images into dimensions similar to the input. Due to the importance of the layer, an optimal number of units search is reported in Subsection 4.4.5.1.

4.4.5.1 Optimal Code layer size

Among the many model's parameters, the main influencing parameters that motivated a search is the size of the encode layer of the CAE. This is also related to the number of output values of the CRF model. Having the speed-up provided by the GPUs for training CSAE, a search for an optimal

size of the code layer is conducted. It is done to reduce arbitrariness in the choice of the number of coding units, and to ensure obtaining the most effective results. Therefore, 100 epochs of CSAE algorithm is run for each of code layer sizes: 8, 10, 20 and 40 units. We started off with 8 units because of its closeness to the presence of two classes in the training data. Then, we allowed more degrees of freedom to see which result demonstrated mostly, the known physical properties of short time bursts while achieving the goals for our training, i.e., selectivity. The results in Figure 4.15 and every other results from this point were also uniformly smoothed with a simple locally weighted moving average filter *MATLAB* function `loess` having a span of 0.10 to arrive at the smoothed lines. Transition threshold described in Subsection 4.4.4 are shown on each plot of Figure 4.15. The transition thresholds with respect to 8, 10, 20 and 40 units at the coding layer are found to be 0.00346, 0.00391, 0.00344 and 0.00574 respectively.

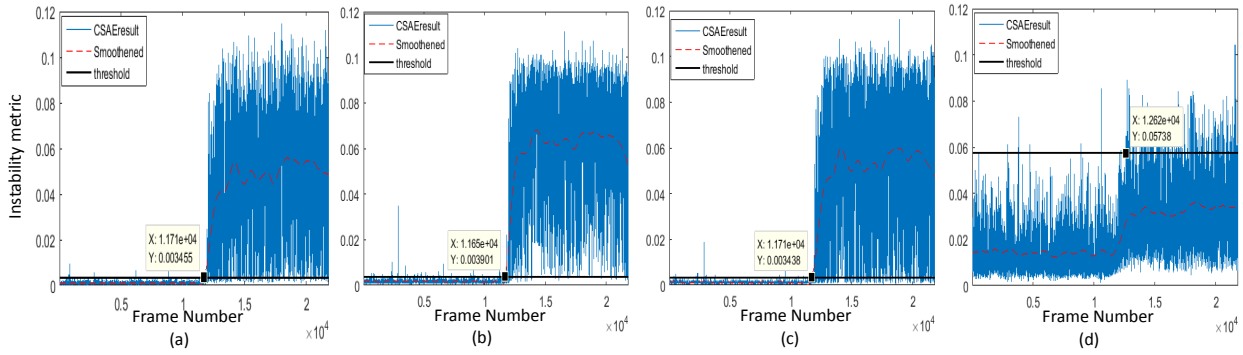


Figure 4.15: Code layer selection for 500_{40to30} with, (a) 8 units, (b) 10 units, (c) 30 units and (d) 40 units.

The results in Figure 4.15 support previously found results (Akintayo et al. (2016b); Sarkar et al. (2015b,c)) of transition stage being between the two regions. It is observed that with 40 units, algorithm does not satisfy the selectivity condition of masking the stable part unlike the other units. This may happen due to the decrease in noise rejection capability with increase in degrees of freedom at the coding layer. Also, the discriminatory ability of the results are assessed. It is a metric that quantifies the maximization of the inter-region separation, while minimizing the intra-region separation similar to a Fisher linear discriminant analysis. However, for result assessment

in this problem, a conservative way is to examine ratio of the variance to the mean provided. The larger the spread around the average, the more the discrimination capability between stable and unstable regions. Therefore, the distribution of z found in Equation 4.5 are also examined on this basis for each of the test protocols.

From the trends of the statistics on Table 4.5, including early signal of the transition shown by the frame #, coding layer with 10 units produced the best results, both visually and statistically. It however fails to be the most discriminatory due to its large mean despite also having the largest variance. We note that performance improves with increase in coding layer length from 8 to 10, while it reduces when the coding layer length is increased further. While an optimal length of the coding layer can be found between 10 and 20, we selected 10 units for performance comparison with SCRF is presented. Transition frame # for 40 units of the layer is not easily found because the validation results are less suppressed compared to the test frame. Hence, in this case early detection may not be feasible.

Table 4.5: CSAE optimum encode layer size metric and transition start frame # for protocol 500_{40to30}.

# of units	μ**(z)**	Σ**(z)**	$\frac{\Sigma(z)}{\mu(z)}$	**frame #**
8	0.0222	0.0238	1.071	11800
10	0.0289	0.0302	1.045	11700
20	0.0241	0.0258	1.072	11800
40	0.0175	0.0246	1.041	$\approx$ 12700

4.4.5.2 CSAE and SCRF comparison

A visual comparison of the distributions of z (Equation 4.5) on test transition protocol, 500_{40to30} via their instability metrics are plotted against frame number for both algorithms. These are shown in Figure 4.16. Clearly, the results of CSAE is more discriminatory in nature, i.e., it has more scatter around its local mean than that of SCRF. CSAE also shows a greater capability than SCRF, to satisfy the training criteria on a new test data set. Therefore, CSAE will be more effective for early

detection of instability. Note, the transition threshold for SCRF as defined in Subsection 4.4.4 is found to be 0.037. On the other hand, threshold for CSAE with 10 code layer units is 0.0040.

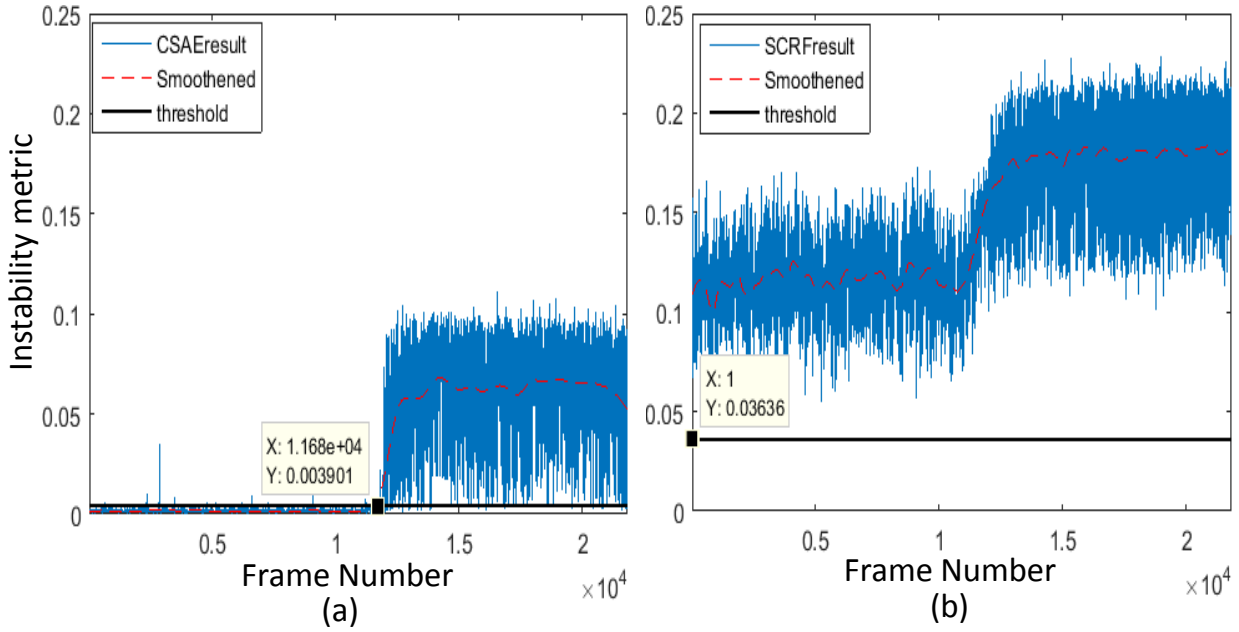


Figure 4.16: Results of, (a) CSAE and (b) SCRF for test transition protocol, 500_{40to30}.

Other differences in computation and memory complexity are shown in Table 4.6. Note that

Table 4.6: CSAE and SCRF comparison

Factor	**CSAE**	**SCRF**
platform	Python	Matlab
model size	$\mathcal{O}$(Mbytes)	$\mathcal{O}$(Kbytes)
CPU inference rate	$\approx 4.7ms/frame$	$\approx 0.24secs/frame$
GPU inference rate	$\approx 1\mu s/frame$	–

with the demonstrated advantage of GPUs, dedicated field programmable gate arrays (FPGAs) built for the proposed CSAE scheme can enable an on-line real-time instability detection tool for real engines.

4.4.6 Early detection

The speed of detection is in terms of the number of frames seen in the stable region before bursts of instability are detected. However, due to the consistency of the CSAE algorithm with our selective training and domain knowledge (i.e., most of the stable frames are suppressed) on the problems analyzed, its results for 4 test transition protocols are shown and discussed in this Subsection.

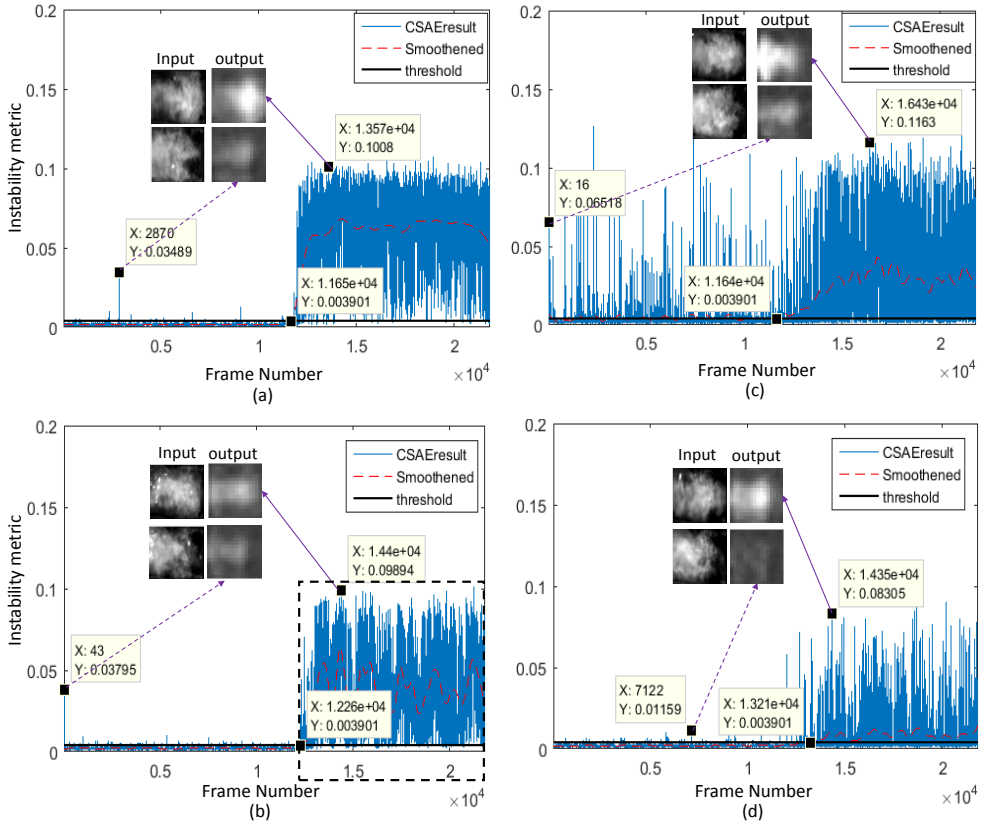


Figure 4.17: Results of transition protocols for, (a) 500_{40to30}, (b) 500_{40to28}, (c) $50_{700to800}$ and (d) $40_{500to600}$ where dashed arrows indicate the results for frames near the unstable flame in the transition region, and thick arrows show results for frames in the supposedly stable regions.

CSAE results on different test transition conditions are presented in Figure 4.17. It shows the capability of the model to suppress stability features of frames in the SR, while revealing some anomalous instability features in the same frames. It also shows the anomalies to be more prominent in the transition regions. Instability metric introduced in Section 4.4.2.2 has been used to evaluate

the strength of each algorithm's ability to mask examples closer to the SR compared to those nearer to the UR. The results are comparable with those found in (Sarkar et al. (2015b)) where the framework used a neural-symbolic approach with a combination of convolutional neural networks and symbolic time series analysis to obtain instability metrics. Note, no background knowledge is provided other than domain knowledge regarding the possibility of short-time instability bursts in the stable regions. Figure 4.17(a) and 4.17(b) have similar transition conditions. Figure 4.17(b) has a leaner mixture and it shows more short term fluctuations in the post-transition phase compared to (a) (as marked by a dotted box in (b)). Furthermore, Figure 4.17(b) signals earlier (at frame 42) the presence of instability compared to Figure 4.17(a) whose first indication is approximately around frame 2870. Also, by considering the underlying physics (Li et al. (2007)) about lean mixtures, the protocol in (c) has the most unstable intermittency in both the SR and the transition phase. It may be considered to be the closest to instability of all the protocols as highlighted in the example frames. In contrast to (c), transition protocol in (d) generally shows results that are closer to stability. This is probably due to the balance provided by its originally richer mixture. It also has the most 'late detection' of the early burst of instability as well as departure from stability among all the protocols.

Table 4.7: Performance metrics and transition start frame # for transition protocols.

	CSAE				**SCRF**			
Protocol	μ**(z)**	Σ**(z)**	$\frac{\Sigma(z)}{\mu(z)}$	**frame #**	μ**(z)**	Σ**(z)**	$\frac{\Sigma(z)}{\mu(z)}$	**frame #**
500_{40to30}	0.0289	0.0302	1.045	11700	0.1449	0.0310	0.214	–
600_{50to35}	0.0266	0.0378	1.422	14500	0.1264	0.0380	0.301	–
500_{40to28}	0.0175	0.0194	1.109	12300	0.1417	0.0330	0.233	–
$50_{700to800}$	0.0142	0.0125	0.881	11700	0.1303	0.0150	0.116	–
$40_{500to600}$	0.0045	0.0033	0.734	13300	0.1262	0.0177	0.141	–

Finally, Table 4.7 shows a summary of the results obtained from the algorithms for all the test transition protocols.

4.4.6.1 Frame labeling

An extension of the algorithm's objectives could be made to implicit labeling. This is achieved by searching through all the frames to detect frames that are adjacent neighbors to a given frame. In clear terms, this means finding the label of a frame given the knowledge of the label of an adjacent flame. This kind of search is usually difficult with most primitive low dimensional local labeling algorithms (e.g. HMM and MEMM) due to dependency depth and 'labeling bias' limitations respectively. For this purpose also, we were motivated to compare the results of CSAE to those of SCRF.

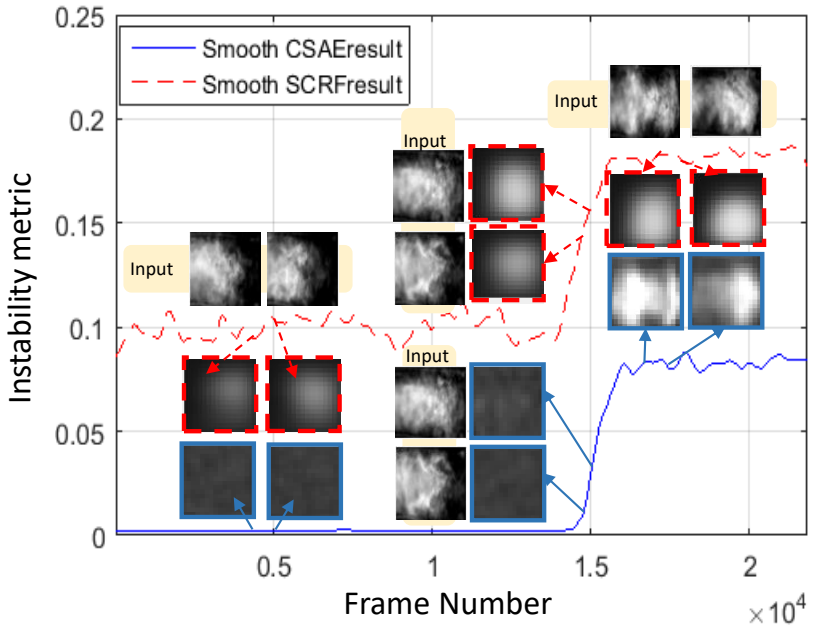


Figure 4.18: Adjacency labeling result of transition protocols for 600_{50to35} at the different regions of the profile while the image frames without any boundaries represent the inputs to the protocols at the points indicated by the arrows.

It was shown that such high dimensional problem can be simplified with values that are calibrated using scales such as our instability metric. The highlighted examples in Figure 4.18 show how labeling may be achieved with the algorithms. Based on qualitative frame-to-frame visualiza-

tion, labels provided by CSAE are shown in Figure 4.18 to outperform that of SCRF. CSAE is able to differentiate labels from frame-to-frame better than the CSAE in the separate flame regions. Frames in the region closest to UR have their the mushroom structures better labeled by CSAE while SCRF does not activate all the units for such labels. Importantly also, we find a gradual transition in the labels of frames in the almost linear transitioning stage of CSAE in much similar way as that of SCRF. It was noted that all input examples used for comparison in the Figure 4.18 were chosen at the similar frame numbers for both algorithms. The results provides briefly the potentials of the algorithm to deriving soft labels from intrinsically labeled classes, two classes in this case.

In summary, the end-to-end convolutional selective autoencoder was deployed for early detection of combustion instabilities in hi-speed flame video. Validation results were performed on data from a laboratory scale swirl-stabilized combustor. In addition to that, the framework was also used to generate fuzzy labels from prior knowledge of hard labeled examples as solution to implicit labeling problem. Conditional random field model results were used to compare the effectiveness of our deep learning based solution approach in both applications. Moreover, CSAE results shown confirm the expert's physical observation when intermmitent coherent structures are present in stable flame regions. Some observed differences in the results are that: (i) CSAE is able to learn and generalize selectivity better than SCRF via more efficient masking the stable region; (ii) Unlike CSAE, SCRF introduces a bias in the instability metric computation for test data, such that its ability to act as an effective filter is hindered; (iii) SCRF succumbs to high false alarm rate during stable combustion. The fact that CSAE can detect instability early for various new (unseen in training phase) protocols while being trained on different protocols shows the generalizability of the proposed algorithm. The results were presented in the light of KL-distance based instability metric that calibrated the closeness to domain knowledge of stable flame frames reproduced by the models. Using the same metric, the architecture was extended for addressing the neighborhood implicit graph labeling problem. The framework can be generalized to soft-labeling of high-dimensional data.

4.5 Conclusion

Feature extraction from other spatial datasets have been approached by improving the denoising autoencoders in Section 3 with state-of-the-art detection convolutional autoencoder. For more tasks generalization of the approach, a selectivity function was designed and incorporated in the convolutional autoencoder. The architecture was thereafter shown on two important applications as a effective candidate for hierarchical feature extraction from this class of datasets.

CHAPTER 5. FEATURE EXTRACTION FROM TEMPORAL DATA – HIERARCHICAL FEATURE EXTRACTION

In this chapter, a proposed hierarchical feature extraction algorithm for unsupervised streaming non-stationary time series (temporal) dataset is presented. The ideas behind the algorithm are methodically and mathematically formulated, and the algorithm is validated on simulated and tested on real life observation.

5.1 Introduction

Feature extraction facilitates effective storage, analysis and transmission of data via high level representations of the raw data (Ralanamahatana et al. (2005)). From the time-series perspective, efficient feature extraction is an important and challenging task in many statistical signal processing and machine learning applications. From a machine learning perspective, time series feature extraction has been considered primarily by two separate communities, involving the parametric and the non-parametric modeling approaches. Examples of parametric models include auto-regressive and/or moving average type models where the number of model parameters remain fixed. The nonparametric techniques on the other hand have the capability to dynamically evolve in the parametric space as novel data features arrive. Some of the popular techniques include variations of the Bayesian Non-Parametric (BNP) strategies such as the Hierarchical Dirichlet Process (HDP) with Hidden Markov Models (Fox et al. (2011)), infinite Hidden Markov Models (Beal et al. (2014)) and variational inference on Bayesian Gaussian mixture models (Blei and Jordan (2006)). However, due to the involvement of latent space evaluation, these techniques tend to be rather slow and thus difficult to be used in on-board, real time applications.

5.2 Background and Motivation

In residential energy disaggregation tasks, it is important determine the energy consumption by various end uses (e.g., appliances or lighting) given the time series of whole building energy usage (Hart (1992); Zeifman and Roth (2011)). Some of the benefits of this analysis are: load balancing, peak demand shaping and the time-of-use pricing. Typically, supervised (Cook et al. (2011); Kolter and Jaakola (2012)) or semi-supervised (Shao et al. (2012)) algorithms are applied to solve such a problem. However, an unsupervised approach for automatically identifying different unique characteristics in whole building electrical energy usage can be particularly useful for this emerging Internet of Things (IoT) application, in terms of easy deployment and enhanced scalability. With this motivation, a Hierarchical Symbolic Dynamic Filtering (HSDF) framework (Akintayo and Sarkar (2015)) is proposed to automatically captures different unique quasi-stationary characteristics in the form of different Probabilistic Finite State Automata (PFSA) models. The entire time series can then be expressed as a higher level PFSA whose states are the PFSAs that model the different unique characteristics. Modeling a quasi-stationary time series as a PFSA is based on the concept of Symbolic Dynamic Filtering (SDF) that approximates a symbolic time series as an observable Markov chain of a certain order (Ray (2004)). However, the quasi-stationary assumption is prohibitive when dealing with the non-stationary characteristics of data. Time series contains multiple quasi-stationary characteristics at a fast time-scale and hence becomes non-stationary when viewed at a slower time-scale. A thorough review of the SDF framework can be found by Ray (2004); Gupta et al. (2008).

Time series signals obtained from dynamical systems can be decomposed into multiple time-scales. Without loss of generality, we consider two such time-scales, namely the fast time-scale at which the acquired signal can be considered to be quasi-stationary and a slow time-scale, at which the time series can be non-stationary in nature. A detailed review of the concept of hierarchical symbolic dynamic filtering (HSDF) using an illustrative example was described in Figure 5.1. A first slow time epoch τ_1 is represented by the PFSA 1 model. Given learning the PFSA 1 from slow time epoch τ_1 (using standard SDF), a second slow time epoch τ_2 is learnt. τ_2 in Figure 5.1

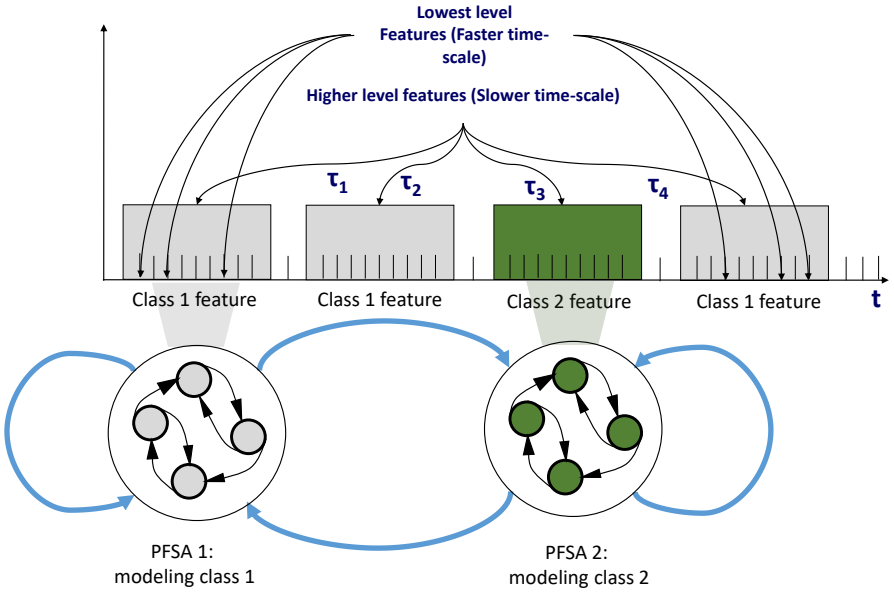


Figure 5.1: Hierarchical PFSA-based feature extraction at a glance

would be considered to be same quasi-stationary characteristics as in τ_1. The challenge was to identify the similarity and classify τ_2 as a member of class 1 represented by PFSA 1. The class retention mechanism can be perceived as a self-transition at a higher logical level where PFSA 1 is considered as a state of the underlying system. Slow time epoch τ_3 however belongs to a new, unforeseen quasi-stationary characteristics. Identifying the change in characteristics from the streaming data and creating new PFSA to represent individual characteristics. At the higher logical layer, this becomes a state transition from state/class 1 (represented by PFSA 1) to state/class 2 (represented by PFSA 2). By starting to learn an upper-level PFSA that model class retentions and transitions. Time series from the underlying system can thus be described in a bottom-up hierarchical manner whereby the goal is to discover multiple PFSAs at the lowest level which in turn become the states of an upper-level PFSA. The concept is related to deep learning architecture where 'features of features' are learned at the each next layer in the hierarchy.

The proposed HSDF scheme ensures transitioning by automatically detecting and modeling unique fast time-scale characteristics via newer PFSA creation. A Multi-scale Symbolic Time Se-

ries Analysis (MSTSA) (Sarkar et al. (2015a)) approach was proposed recently for characterizing seismic activities monitored by Unattended Ground Sensor (UGS) in an online manner. The key difference between the MSTSA approach and the hierarchical feature extraction proposed is that no label or supervision (i.e., without knowing how many unique characteristics or classes are present in data and when or how often they occur) is considered. The proposed modeling architecture is also notionally similar to that of the Switching Linear Dynamical Systems (SLDS) (Fox et al. (2009)) While SLDS is linear, nonlinear modeling for capturing the different unique characteristics was considered. Thereafter, the algorithm is validated on time series data from a nonlinear dynamical systems modeled by the chaotic Duffing and Van der Pol systems (Rao et al. (2009a)). The algorithm was tested on the publicly available Reference Energy Disaggregation Dataset (REDD) for residential electrical energy disaggregation.

The main contributions of this research are: The development of a novel Hierarchical Symbolic Dynamic Filtering (HSDF) algorithm that can model the non-stationary time series data comprised of several quasi-stationary behavior where neither the behaviors nor the number of unique characteristics are known; The demonstration of the effects of various concepts such as adaptive Chinese Restaurant Process (CRP), likelihood change rate and stickiness adjustment on the HSDF performance; The development of an off-line PFSA model revision strategy to improve HSDF performance; Testing and validation of the proposed algorithm, as well as performance comparison with SLDS (realized via the sticky HDP-HMM approach) in order to show that the proposed algorithm is a faster and computationally more efficient method with vast applicability in real-life time series feature extraction.

A description of SDF via the proposal's other main constituents (namely, CRP and stickiness factor) are presented in Subsection 2.7.2.

5.2.1 The CRP distribution and stickiness adjustment

This subsection briefly describes a couple of basic statistical concepts used in the proposed formulation, namely the Chinese Restaurant Process (CRP) and data likelihood adjustment using the stickiness factor. Recently, these ideas have been extensively used in nonparametric modeling techniques and therefore details can be found in the related literature (Teh (2007); Fox et al. (2011)).

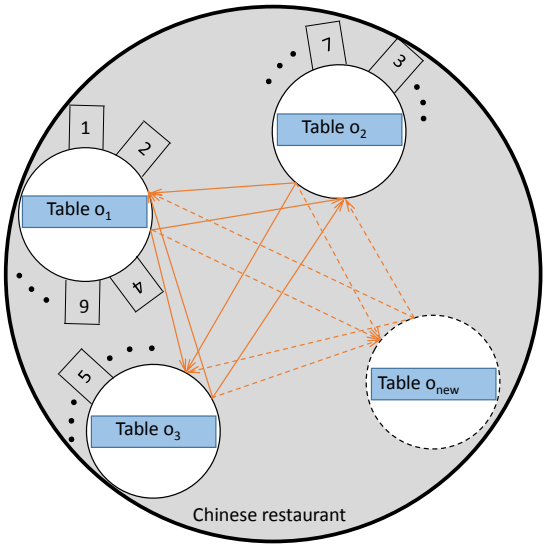


Figure 5.2: Illustration of Chinese Restaurant Process with numbers: 1,2,3, $\cdots$ indicating customers' order of arrival.

CRP represents a discrete sequence over partitions that is suitable for modeling infinite mixtures, hence often used for modeling clusters in Bayesian frameworks. CRP shares some similarities with the stick breaking and the Dirichlet Process, but with some subtle differences in how the processes evolve (Teh (2007)). The crux of the CRP distribution is to model the tendencies of newly arriving customers to a fictional Chinese restaurant to either sit in an existing table $\in O$ or in a new table, o_{new} (Aldous (1985)) with less restriction on number of tables or customers at a table (as

illustrated in Figure 5.2). Therefore, CRP is a suitable candidate for nonparametric modeling. The CRP distribution can be mathematically described as follows:

$$Pr_{\epsilon}(o \in O) = \frac{\mathbb{C}(o)}{[\sum_{x \in O} \mathbb{C}(x)] + \epsilon} \tag{5.1}$$

$$Pr_{\epsilon}(o_{new}) = \frac{\epsilon}{[\sum_{x \in O} \mathbb{C}(x)] + \epsilon} \tag{5.2}$$

where $\mathbb{C}(\cdot)$ signifies a concentration function and ϵ is called the CRP parameter. Originally, a CRP process is used to determine whether a new PFSA model is required to model a newly arriving slow time epoch or an existing PFSA would suffice.

While induction of CRP can help in deciding the need for a new PFSA model, noise and spurious disturbance present in real data can drive the decision system to instability. That is, many unnecessary new PFSA models may get generated and the decision may then fluctuate among different PFSA models that are close to each other, with closeness based on an appropriate metric. Similar situation arises in other unsupervised techniques as well such as the HDP-HMM. Assuming inference could be made after the arrival of several slow time epochs, an off-line revision that is described in Subsection 5.3.4 will be effective at merging such spurious classes. However, in most real-life application, decisions have to be taken in an online manner, i.e., soon after the arrival of the new slow time epochs. A stickiness factor described by Fox et al. (2011); Akintayo and Sarkar (2015) was found to be effective for reducing such fluctuations. The extra information incorporated by stickiness is to lean slightly towards the class occupied by the most recently assigned (before the current epoch) slow time epoch. The ideas discussed in this section are described mathematically in Section 5.3.

5.3 Proposed Hierarchical SDF Methodology

The proposed hierarchical symbolic dynamic filtering (HSDF) framework, along with the learning scheme using streaming non-stationary time series data are described in this section. In general,

two technical challenges are involved in such a problem involving streaming data. The first is reliability (or accuracy) of the inference for decision-making, while the second is how quickly the inference can be made. The aim of the research was to present a comprehensive approach to tackling both challenges for robust decision-making. Most real-life dynamical systems, especially those with safety, security, reliability or dependability concerns (Sarkar et al. (2016, 2008); Gupta et al. (2008); Jin et al. (2011); Sarkar et al. (2013a); Chakraborty et al. (2008b)) require such online inference capability for decision-making. We begin with a mathematical derivation of the data likelihood of a newly arriving slow time epoch given for a few existing classes (PFSA models) described in Subsection 2.7.2. Formulation of an adaptive CRP process and stickiness factor for assigning slow time epochs to existing and new classes are provided in Subsection 5.3.2. Finally, the outline of an off-line algorithm for periodic revision (over a few slow time epochs) of the space of PFSA models learnt by the online process is presented in Subsection 5.3.4.

5.3.1 Data Likelihood estimation

Let us assume that K classes representing K unique quasi-stationary behavior occurred in the past data epochs have already been identified. Let the distinct set of classes be $\{C^i : i = 1, 2, \cdots, K\}$, over the same sets of symbol Ξ and state Θ, and each class C^i is modeled by a PFSA $\mathcal{P}^i = (\Theta^i, \Xi, \delta^i, \Omega^i)$. Also, let symbol strings belonging to the class be $S^i \triangleq s_1^i s_2^i \ldots$. An appropriate depth D is selected for the D-Markov machine from which the morph (probability) matrix Ω^i has been derived. Each row of the Ω is normalized in order to perform inference on a new slow time epoch.

Let the m^{th} row of Ω^i be denoted as Ω_m^i and the n^{th} element of the m^{th} row as $\Omega_{mn}^i \geq 0$ and $\sum_{n=1}^{|\Xi|} \Omega_{mn}^i = 1$. The *a priori* probability density function $f_{\Omega_m^i|S^i}$ of the random row-vector Ω_m^i that is conditioned on a symbol string S^i can be modeled by the Dirichlet distribution (Ferguson (1973); Sethuraman (1994)) as,

$$f_{\Omega_m^i|S^i}(\boldsymbol{\omega}_m^i|S^i) = \frac{1}{B(\boldsymbol{\alpha}_m^i)} \prod_{n=1}^{|\Xi|} (\omega_{mn}^i)^{\alpha_{mn}^j - 1} \tag{5.3}$$

where, each column of the Ω^i is represented by $\boldsymbol{\omega}_m^i$ as,

$$\boldsymbol{\omega}_m^i = \begin{bmatrix} \omega_{m1}^i & \omega_{m2}^i & \dots & \omega_{m|\Xi|}^i \end{bmatrix}$$

and the constant for normalization is

$$B(\boldsymbol{\alpha}_m^i) \triangleq \frac{\prod_{n=1}^{|\Xi|} \Gamma(\alpha_{mn}^i)}{\Gamma(\sum_{n=1}^{|\Xi|} \alpha_{mn}^i)} \tag{5.4}$$

where, $\Gamma(\bullet)$ denotes the gamma function, and $\boldsymbol{\alpha}_m^i = \begin{bmatrix} \boldsymbol{\alpha}_{m1}^i & \boldsymbol{\alpha}_{m2}^i & \dots & \boldsymbol{\alpha}_{m|\Xi|}^i \end{bmatrix}$ with

$$\alpha_{mn}^i = N_{mn}^i + 1 \tag{5.5}$$

At a state θ_m, the number of times the symbol ξ_n arises in the transition to a new state is modeled by N_{mn}^i as,

$$N_{mn}^i \triangleq \left| \{(s_k^i, v_k^i) : s_k^i = \xi_n, v_k^i = \theta_m\} \right| \tag{5.6}$$

where s_k^i is the k^{th} symbol in S^i and v_k^i is the k^{th} state as obtained from the symbol sequence S^i. Note that a state is defined as a sequence of past D symbols. $N_m^i \triangleq \sum_{n=1}^{|\Xi|} N_{mn}^i$ computes the number that state θ_m occur in the state sequence. From Equation 5.4 and 5.5, it follows that

$$B(\boldsymbol{\alpha}_m^i) = \frac{\prod_{n=1}^{|\Xi|} \Gamma(N_{mn}^i + 1)}{\Gamma(\sum_{n=1}^{|\Xi|} N_{mn}^i + |\Xi|)} = \frac{\prod_{n=1}^{|\Xi|} (N_{mn}^i)!}{(N_m^i + |\Xi| - 1)!} \tag{5.7}$$

where the standard definition, $\Gamma(n) = (n-1)! \ \ \forall n \in \mathbb{N}$ has been used.

Markov property of the $PFSA\ \mathcal{P}^i$, ensures that the $(1 \times |\Xi|)$-dimension row vectors of Ω^i are statistically independent of each other, $\forall m = 1, \dots |\Theta|$. Equations 5.4 and 5.7 thus lead to conditioning of the *a priori* joint density, $f_{\Omega^i|S^i}$ of the probability morph matrix, Ω^i on the symbol string, S^i as,

$$\begin{aligned} f_{\Omega^i|S^i}(\boldsymbol{\omega}^i|S^i) &= \prod_{m=1}^{|\Theta|} f_{\Omega_m^i|S^i}\left(\boldsymbol{\omega}_m^i|S^i\right) \\ &= \prod_{m=1}^{|\Theta|} \left(N_m^i + |\Xi| - 1\right)! \prod_{n=1}^{|\Xi|} \frac{(\boldsymbol{\omega}_m^i)^{N_{mn}^i}}{(N_{mn}^i)!} \end{aligned} \tag{5.8}$$

where, $\boldsymbol{\omega}^i = \begin{bmatrix} (\boldsymbol{\omega}_1^i)^T & (\boldsymbol{\omega}_2^i)^T & \cdots & (\boldsymbol{\omega}_{|\Theta|}^i)^T \end{bmatrix} \in [0,1]^{|\Theta| \times |\Xi|}$, and T is a transpose operator.

At this point, if a new slow time test epoch is represented by $\widetilde{S}$, its probability of belonging to a certain PFSA model $(\Theta, \Xi, \delta, \Omega^i)$, given the morph matrix Ω^i derived from the training symbol sequence S^i can be represented as a product of multinomial distributions (Wilks (1963)) as,

$$\begin{aligned} & \Pr\left(\widetilde{S}|\Theta, \delta, \Omega^i\right) \\ &= \prod_{m=1}^{|\Theta|} (\widetilde{N}_m)! \prod_{n=1}^{|\Xi|} \frac{\left(\Omega^i_{mn}\right)^{\widetilde{N}_{mn}}}{(\widetilde{N}_{mn})!} \qquad (5.9) \\ &\triangleq \Pr\left(\widetilde{S}|\Omega^i\right) \quad \text{as } \Theta \text{ and } \delta \text{ are kept invariant} \qquad (5.10) \end{aligned}$$

where, given a state θ_m, the number of times the symbol ξ_n present in the testing string $\widetilde{S}$ occurs during transition to a new state is modeled by $\widetilde{N}_{mn}$ as,

$$\widetilde{N}_{mn} \triangleq \left|\{(\tilde{s}_k, \tilde{v}_k) : \tilde{s}_k = \xi_n, \tilde{v}_k = \theta_m\}\right| \qquad (5.11)$$

where again, the k^{th} symbol in the observed string $\widetilde{S}$ is $\tilde{s}_k$, and the k^{th} state derived from $\widetilde{S}$ is denoted by $\tilde{v}_k$.

Now, Equation 5.8 and 5.9 can be combined to obtain the probability of a symbol string $\widetilde{S}$ belonging to a class characterized by already observed symbol string S^j. With the derivation presented by Sarkar et al. (2000), the following conditional distribution was obtained,

$$\begin{aligned} \mu(\widetilde{S}|S^i) = & \prod_{m=1}^{|\Theta|} \frac{(\widetilde{N}_m)!\left(N^i_m + |\Xi| - 1\right)!}{\left(\widetilde{N}_m + N^i_m + |\Xi| - 1\right)!} \\ & \times \prod_{n=1}^{|\Xi|} \frac{(\widetilde{N}_{mn} + N^i_{mn})!}{(\widetilde{N}_{mn})!(N^i_{mn})!} \qquad (5.12) \end{aligned}$$

where $\widetilde{N}_m \triangleq \sum_{n=1}^{|\Xi|} \widetilde{N}_{mn}$.

In practice, Stirling's approximation for the logarithm of a factorial $\log(n!) \approx n\log(n) - n$ Pathria (1996) is mostly easier to compute, especially when either (or both) of N^i and $\widetilde{N}$ consist of statistically large enough sample points (but still not be enough to directly estimate a Π matrix at the testing phase). At this point, the likelihood probability, $\Pr(\widetilde{S}|S^i)$ may be easily found by normalizing the conditional factors in Equation 5.12.

5.3.2 Assignment of a slow time-scale behavior to an existing or new class

The objective of the inference process is to compute the probability of assigning a slow time epoch τ_j to a class $C^i \in \mathbf{C}$ (where $\mathbf{C} = \{C^i \ \forall \ i = 1, \cdots, K\}$ is the set of existing classes) or a newly created class C^{K+1}. Let the symbol sequence for the current slow time epoch be $\widetilde{S}_{\tau_j}$. Then the likelihood for class C^i for current epoch τ_j is given by $\mu(\widetilde{S}_{\tau_j}|S^i)$ as described in the previous subsection.The posterior probability for class selection can be denoted by $\Pr(C^i, S^i|\widetilde{S}_{\tau_j})$ which is equivalent to $\Pr(C^i|\widetilde{S}_{\tau_j})$ in this case since all existing classes are completely characterized by symbol sequences $S^i \ \forall i$.

With this setup, we obtain the following:

$$\Pr(C^i|\widetilde{S}_{\tau_j}) \propto \mu(\widetilde{S}_{\tau_j}|S^i) \ \ \forall i = 1, \cdots, K \tag{5.13}$$

We use the Chinese Restaurant Process (CRP) to introduce the likelihood of a new class C^{K+1} with a CRP hyperparameter γ_j as follows (note that the hyperparameter is specific to the test epoch τ_j).

$$\mu_{\gamma_j}(C^{K+1}|\widetilde{S}_{\tau_j}) = \gamma_j \sum_{i=1}^{K} \mu(\widetilde{S}_{\tau_j}|S^i) \Rightarrow \sum_{i=1}^{K} \mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j}) = (1-\gamma_j) \sum_{i=1}^{K} \mu(\widetilde{S}_{\tau_j}|S^i) \tag{5.14}$$

CRP hyperparameter γ_j (that was described in Equation 5.2) is given by the following expression.

$$\gamma_j = \frac{\epsilon}{\left[\sum_{i=1}^{K} \mu(\widetilde{S}_{\tau_j}|S^i)\right] + \mathbf{b}\epsilon} \tag{5.15}$$

where, $\epsilon \geq 0$ is a real valued parameter and $\mu(\widetilde{S}_{\tau_j}|S^i)$ is treated as the concentration or strength function found in Equation 5.1 and 5.2 for the CRP formulation. However, instead of the classical formulation Akintayo and Sarkar (2015), we introduce a new scalar multiplier $\mathbf{b}$ that modifies the likelihood of creating a new class. The choice of $\mathbf{b}$ will depend on a parameter $\mathbf{A}(\Delta, i)$ that captures the rate of change of data likelihood as follows:

$$\mathbf{A}(\Delta, i) = \frac{1}{\Delta} \sum_{p=1}^{\Delta} \left[\mu(\widetilde{S}_{\tau_{j-p}}|S^i) - \mu(\widetilde{S}_{\tau_j}|S^i)\right] \tag{5.16}$$

where Δ is a memory parameter that accommodates likelihoods of past epochs. It is evident from the expression of $\mathbf{A}(\Delta, i)$ that it is essentially an expected reduction of likelihood of exisiting classes at the current epoch τ_j. A high value of Δ reduces the noise in estimation, which can also reduce the senstivity to class changes. While $\mathbf{A}(\Delta, i)$ can be incorporated in various ways to compute the CRP parameter, it is accommodated in a discrete manner in the present formulation. Note that the condition with high values of $\mathbf{A}(\Delta, i)\ \forall i$ suggests a significant drop in likelihood of all existing classes which increases the possibility of a new class generation. Therefore, a positive threshold, ν is chosen such that when $\mathbf{A}(\Delta, i) > \nu\ \forall i$, we use $\mathbf{b} = 1$ (i.e., classical formulation). Otherwise, we reduce the possibility of new class generation by taking $\mathbf{b} = 2$ for the adaptive CRP formulation.

At this point, we introduce the notion of 'stickiness' in our proposed algorithm which is based on the fact that a real-life system usually may not fluctuate its operating point or internal parametric condition at each slow time epoch. In the present context, this means that if a slow time epoch τ_{j-1} belongs to a class, $C^k \in \mathbf{C}$, then there will be a high likelihood for new streaming data at epoch τ_j to belong to C^k as well. This notion is incorporated into the formulation by introducing a positive bias towards the last seen class C^k as follows:

$$\mu_{\gamma_j}(C^k|\widetilde{S}_{\tau_j}) = \max\left\{\frac{\kappa}{1-\kappa}\sum_{i=1}^{K+1}\mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j}), \mu_{\gamma_j}(C^k|\widetilde{S}_{\tau_j})\right\} \tag{5.17}$$

where $0 < \kappa < 1$ is the stickiness factor. Note, the rationale behind this adjustment is to ensure a certain minimum likelihood for the last seen class C^k and in this context, the proposed formulation ensures that

$$\frac{\mu_{\gamma_j}(C^k|\widetilde{S}_{\tau_j})}{\sum_{i=1}^{K+1}\mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j})} \geq \kappa \tag{5.18}$$

This can be proved by considering the extreme case when $\mu_{\gamma}(C^k|\widetilde{S}_{\tau_j}) = 0$, before applying the stickiness factor. Numerically, the 'stickiness' adjustment significantly reduces the 'hunting behavior' in class identification and creation process which will be demonstrated via numerical simulation results in the next section.

Finally, the $\mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j})$ factors are normalized to obtain the posterior probabilities $\Pr(C^i|\widetilde{S}_{\tau_j})$ for each class as follows:

$$\Pr(C^i|\widetilde{S}_{\tau_j}) = \frac{\mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j})}{\sum \mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j})} \tag{5.19}$$

We generate a random sample from this distribution to take a decision of class identification and generation at the testing epoch τ_j.

The online algorithm for class assignment is summarized in Algorithm 1. Note, we assume that partitioning and state construction are already performed before we begin the following algorithm.

Hence, the alphabet Ξ and state set Θ and the corresponding indices n and m are already defined.

Input Parameters: CRP parameter ϵ, memory parameter Δ,

likelihood rate threshold ν and stickiness parameter κ

Data Input: Slow time epochs: $\tau_1, \tau_2, \cdots$ of symbolized string segments $\widetilde{S}_{\tau_l}$

Initialization: $\mathbf{C} = \{C^1\}$ and Compute N_{mn}^1 using $\widetilde{S}_{\tau_1}$

forall $\tau_2, \tau_3, \cdots$ **do**

Compute $\widetilde{N}_{mn}$ using $\widetilde{S}_{\tau_j}$

if $j < \Delta + 1$ **then**

Compute γ_j using Equation 5.15 with $\mathbf{b} = 2$

else

Compute $\mathbf{A}(\Delta, i)$ for all existing classes C^i using Equation 5.16

if $\boldsymbol{A}(\Delta, i) > \nu \ \forall i$ **then**

Compute γ_j using Equation 5.15 with $\mathbf{b} = 1$

else

Compute γ_j using Equation 5.15 with $\mathbf{b} = 2$

end

end

Compute $\mu_{\gamma_j}(C^i|\widetilde{S}_{\tau_j})$ using Equation 5.14

$\forall C^i \in \mathbf{C} = \{C^1, C^2, \ldots, C^K\}$ and C^{K+1}

Apply 'stickiness' adjustment using Equation 5.17

Compute $\Pr(C^i|\widetilde{S}_{\tau_j})$ $\forall i \in \{1, 2, \ldots, C^{K+1}\}$ using Equation 5.19

Assign $\widetilde{S}_{\tau_j}$ to a class via sampling from the distribution $\Pr(C^i|\widetilde{S}_{\tau_j})$

if $j \in \{1, 2, \ldots, K\}$ **then**

Update N_{mn}^j by appending $\widetilde{S}_{\tau_l}$ to S^j

else if $j = K + 1$ **then**

Update $\mathbf{C}$ as $\{C^1, C^2, \ldots, C^K, C^{K+1}\}$

Compute N_{mn}^{K+1} using $\widetilde{S}_{\tau_l}$

end

end

Algorithm 1: Online HSDF algorithm

5.3.3 Computational complexity of HSDF

The computation involved in getting the symbols from the continuous time series given the discretization parameters is $\mathcal{O}(T)$, where T is the number of data points in the time series. The core PFSA formulation in the feature extraction model is an observable Markov model where the state transition matrix can be updated simply with a frequency-based scheme. Therefore, with a fixed algebraic structure of the PFSA (i.e., with fixed number of states and fixed alphabet), the PFSA estimation also needs $\mathcal{O}(T)$ computation.

The worst case computation is however incurred during the likelihood computation for different PFSA models (corresponding to different existing classes) for a given test epoch. In a strict implementation, this step involves a few factorial computations which becomes extremely computation intensive as T grows. However, we use the Stirling's approximation such that the computation does not grow with the number of datapoints. If the number of existing classes is fixed, then the complexity of this step grows linearly with the number of epochs which is basically proportional to T if the epoch length is kept constant. As the same computation has to be repeated for all existing classes we obtain that with a fixed epoch length and a maximum possible number of classes K, the computational complexity becomes $\mathcal{O}(KT)$. Therefore, the overall complexity of the online HSDF algorithm also becomes $\mathcal{O}(KT)$.

5.3.4 Off-line PFSA revision

Algorithm 1 operates at the lowest logical layer in an online manner for learning multiple PFSA models representing different unique quasi-stationary characteristics. Representation of these characteristics is performed by considering the changes in the data likelihood and its rate of change. However, when the data quality is low especially in terms of signal-to-noise ratio (SNR), the online learning algorithm may generate many spurious classes. In such cases, redundant PFSA models may be pruned periodically, that is, after a few slow time epochs have been observed. The pruning step proposed here merges different PFSA models whose proximity are evaluated with the metric laid out in the Definition 5.3.1 below (according to Mukherjee and Ray (2014)).

Definition 5.3.1. (Distance Metric for PFSA) Let $\mathcal{P}^1 = (\Theta^1, \Xi, \delta^1, \Omega^1)$ and $\mathcal{P}^2 = (\Theta^2, \Xi, \delta^2, \Omega^2)$ be two PFSA with a common alphabet Ξ. Let $Pr_1(\Xi_r)$ and $Pr_2(\Xi_r)$ be the steady state probability vectors of generating words of length r from the PFSA, $\mathcal{P}^1$ and $\mathcal{P}^2$, respectively, i.e., $Pr_1(\Xi_r) \triangleq [Pr(w)]_{w \in \Xi_r}$ for $\mathcal{P}^1$ and $Pr_2(\Xi_r) \triangleq [Pr(w)]_{w \in \Xi_r}$ for $\mathcal{P}^2$. Then, the metric for the distance between the PFSA models, $\mathcal{P}^1$ and $\mathcal{P}^2$ is defined as

$$\Phi(\mathcal{P}^1, \mathcal{P}^2) \triangleq \lim_{n \to \inf} \sum_{r=1}^{n} \frac{\| Pr_1(\Xi_r) - Pr_2(\Xi_r) \|_{l_1}}{2^{r+1}} \tag{5.20}$$

where the norm $\| \star \|_{l_1}$ indicates the sum of absolute values of the elements in the vector $\star$.

Thus, the pruning step can merge two different PFSA models identified by online HSDF, $\mathcal{P}^1$ and $\mathcal{P}^2$ when $\Phi(\mathcal{P}^1, \mathcal{P}^2) < \eta$, where $\eta > 0$. The metrics utilized have been evaluated on symbols whose word length are 1. Note that this revision step can be considered to be part of an off-line process for learning the higher-level (Tier 2) PFSA.

5.4 Improvement of Data Likelihood

The algorithm proposed in this section inherently aims to maximize the data likelihood as the CRP formulation uses the likelihood of all the existing classes as concentration or strength function at any given epoch. When the likelihood values of the existing classes drop significantly, a new class is created to keep the data likelihood high with respect to the overall hierarchical model. Likelihood visualization in Section 5.5 supports this notion as well. We observe that this process is equivalent to minimizing the Kullback-Liebler (KL) Divergence (Kullback and Liebler (1951)). Similar observations were made by Roux and Bengio (2008); Bengio (2009). Therefore, KL Divergence $\to 0$ can be a relevant objective to learn the proposed hierarchical model and hence can be used for assuring that the algorithm can converge. Also, note that the stickiness adjustment and the PFSA revision step aims to reduce the number of PFSA at the lower layer without significant loss in data likelihood. Hence, the overall algorithm aims to minimize model complexity while capturing data characteristics (likelihood).

Before demonstrating the equivalence between data likelihood and KL Divergence stated above, we present some mathematical preliminaries.

5.4.1 Preliminaries

The gamma function $\Gamma(\alpha)$ can be expressed as,

$$\Gamma(\alpha) \triangleq e^{-\alpha}\alpha^{\alpha-\frac{1}{2}}\sqrt{2\pi}(1+\frac{1}{12\alpha}+\mathcal{O}(\frac{1}{\alpha^2})) \tag{5.21}$$

Using Stirling's approximation, the expression can be simplified under the assumption of $\alpha \gg \frac{1}{2}$ as the following:

$$\Gamma(\alpha) \approx e^{-\alpha}\alpha^{\alpha} \tag{5.22}$$

Using this formula, we can rewrite the normalizing constant described in our online classification approach (see Equation 5.4) as

$$B(\boldsymbol{\alpha}_m^i) \approx \frac{\prod_{n=1}^{|\Xi|} e^{-\alpha_{mn}^i}(\alpha_{mn}^i)^{\alpha_{mn}^i}}{e^{-(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)}(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)^{(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)}} \tag{5.23}$$

By eliminating common terms from the numerator and the denominator, we obtain

$$B(\boldsymbol{\alpha}_m^i) \approx \frac{\prod_{n=1}^{|\Xi|}(\alpha_{mn}^i)^{\alpha_{mn}^i}}{(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)^{(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)}} \tag{5.24}$$

Let the constant denominator term $(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)^{(\sum_{n=1}^{|\Xi|}\alpha_{mn}^i)}$ be denoted as $Z = (N_m^i + |\Xi|)^{N_m^i+|\Xi|}$. With this setup Equation 5.8 in Section 5.3.1 can be rewritten as

$$f_{\Omega^i|S^i}(\boldsymbol{\omega}^i|S^i) \approx \prod_{m=1}^{|\Theta|} Z \prod_{n=1}^{|\Xi|} \frac{(\boldsymbol{\omega}_m^i)^{N_{mn}^i}}{(N_{mn}^i+1)^{N_{mn}^i+1}} \tag{5.25}$$

Similarly, at the testing stage, Equation 5.10 can be rewritten as

$$\Pr\left(\widetilde{S}|\Omega^i\right) \approx \prod_{m=1}^{|\Theta|} \widetilde{Z} \prod_{n=1}^{|\Xi|} \frac{(\Omega_{mn}^i)^{\widetilde{N}_{mn}}}{(\widetilde{N}_{mn}+1)^{\widetilde{N}_{mn}+1}} \tag{5.26}$$

where $\widetilde{Z} = (\widetilde{N}_m)^{\widetilde{N}_m}$.

Theorem 5.4.1. *At a testing epoch, maximizing the loglikelihood* $\log \Pr\left(\widetilde{S}|\Omega^i\right)$ *is equivalent to minimizing KL Divergence between the testing data distribution and training data distribution.*

Proof sketch:

$$\log \Pr\left(\widetilde{S}|\Omega^i\right) = \sum_{m=1}^{|\Theta|}\left(\log \widetilde{Z} - \sum_{n=1}^{|\Xi|}\left(\left[\widetilde{N}_{mn}+1\right]\log\left[\widetilde{N}_{mn}+1\right] - \widetilde{N}_{mn}\log\Omega^i_{mn}\right)\right) \tag{5.27}$$

After some algebraic rearrangement we obtain,

$$\log \Pr\left(\widetilde{S}|\Omega^i\right) = \sum_{m=1}^{|\Theta|}\left(-\sum_{n=1}^{|\Xi|}\left(\left[\widetilde{N}_{mn}+1\right]\log\left[\frac{\widetilde{N}_{mn}+1}{\Omega^i_{mn}}\right]\right)\right) + \sum_{m=1}^{|\Theta|}\left(\log(\widetilde{Z}) - \sum_{n=1}^{|\Xi|}\log\Omega^i_{mn}\right) \tag{5.28}$$

Note that $\sum_{m=1}^{|\Theta|}\log(\widetilde{Z})$ is a constant normalization factor term and Ω^i_{mn} represents the models learnt at the training stage for the existing classes (hence, does not change significantly). Therefore, if we aim to maximize the log-likelihood $\log \Pr\left(\widetilde{S}|\Omega^i\right)$ over all models (or classes) denoted by the index i, we obtain

$$\begin{aligned}\arg\max_i \log \Pr\left(\widetilde{S}|\Omega^i\right) &\approx \arg\max_i \sum_{m=1}^{|\Theta|}\left(-\sum_{n=1}^{|\Xi|}\left(\left[\widetilde{N}_{mn}+1\right]\log\left[\frac{\widetilde{N}_{mn}+1}{\Omega^i_{mn}}\right]\right)\right) \\ &= \arg\max_i \sum_{m=1}^{|\Theta|}\left(-\sum_{n=1}^{|\Xi|}\left([\widetilde{\alpha}_{mn}]\log\left[\frac{\widetilde{\alpha}_{mn}}{\Omega^i_{mn}}\right]\right)\right) = \arg\min_i \left(KL \text{ Divergence}\left(\widetilde{\alpha}, \Omega^i\right)\right)\end{aligned} \tag{5.29}$$

where $\widetilde{\alpha}$ represents the distribution of the testing data and Ω^i represents the training data distribution (for model i).

5.5 Validation Results and Discussion

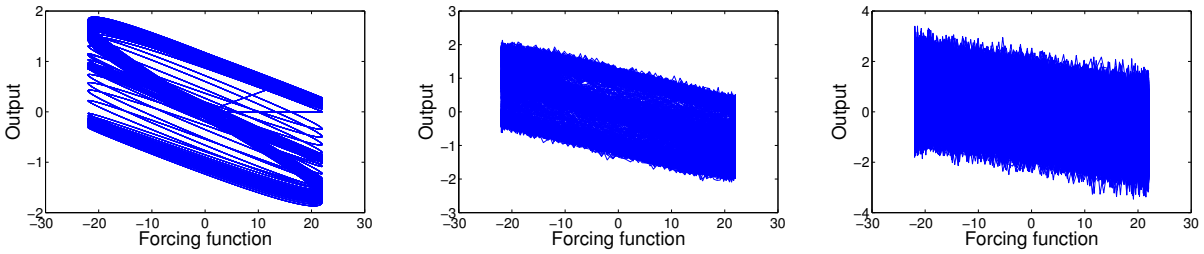
The proposed algorithm is tested and validated in this section using data from simulated switching nonlinear dynamical systems. We begin with describing the simulation system and data generation scheme for validation.

5.5.1 Simulated nonlinear dynamical systems

The chaotic Duffing system whose mathematical model is described in Equation 5.30, which is a popular choice as a nonlinear system (Rao et al. (2009a)) was simulated.

$$\frac{d^2x(t)}{dt^2} + \beta\frac{dx(t)}{dt} + \alpha_1 x(t) + \lambda x^3(t) = Acos(\mathbf{w}t) \tag{5.30}$$

where $A = 22.0$ is the input amplitude, $\mathbf{w} = 5.0$ rad/s is its frequency of excitation, excitation harmonics, $\alpha_1 = 1.0$, stiffness, $\lambda = 1.0$. It is know that varying β, the dissipation parameter causes change in the system behavior and a sudden shift or bifurcation occurs around $\beta = 0.3$ (Rao et al. (2009b)). Hence, $\beta = 0.1$ signifies an operating region before bifurcation and $\beta = 0.4$ represents a system behavior after bifurcation. Therefore, a non-stationary time series with two types of quasi-stationary segments can be generated by randomly selecting between the two β values for different segments. Plots of the output x vs. the forcing function are shown in Figure 5.3 under different noise contamination levels.



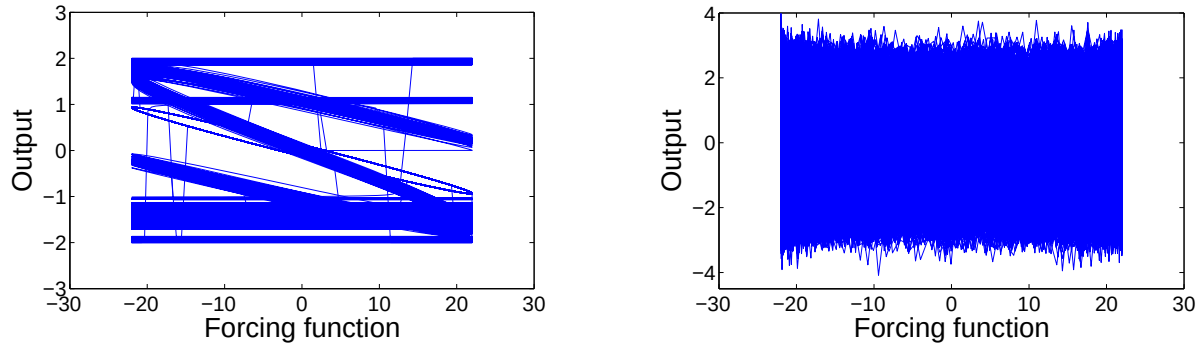
(a) (b) (c)

Figure 5.3: Input-output plots of non-stationary dynamics for chaotic Duffing System under various signal to noise ratio (SNR) - Plate (a) SNR = ∞, Plate (b) SNR = 9 and Plate (c) SNR = 1.

Also three types of quasi-stationary segments or features are generated by adding a Van der Pol oscillation system behavior which is given by Tsatsos (2006) to the 2-class problem simulated by the Duffing system.

$$\frac{d^2x(t)}{dt^2} + 1000x^2(t)\frac{dx(t)}{dt} + x(t) = 1000 \tag{5.31}$$

Figure 5.4 shows the plots of the output x vs. the forcing function with all the three features (two from the Duffing system and one from the Van der Pol system) under two different noise contamination levels. For both 2-features and 3-features cases, time series data with randomly generated 400 epochs (with 1000 data points of one particular feature in each epoch) are used for testing. Also for symbolic dynamic analysis, the raw time series is symbolized with a uniform partitioning (i.e., equal width binning) into 7 bins that is found to be sufficient experimentally for most cases.



(a) (b)

Figure 5.4: Input-output plots of non-stationary dynamics for a random mix of chaotic Duffing system and Van der Pol system under various signal to noise ratio (SNR) - Plate (a) SNR = ∞, Plate (b) SNR = 1.

5.5.2 Results and performance comparison

The performance of the proposed algorithm is evaluated on both the two and three features test cases beginning with analysis to explore the effects of adaptive CRP formulation with parameter **b**, stickiness adjustment and off-line revision steps.

Performance with classical CRP formulation: Many traditional methods only consider data likelihood for feature change detection or outlier detection purposes. However, the algorithm here utilized the change in data likelihood to decide on generating new models. It incorporates the effect of change in likelihood via using the adaptive CRP formulation with parameter **b**. While in our algorithm, **b** can take a value of 1 or 2 depending on the parameter $\mathbf{A}(\Delta, i)$ (Δ is chosen

to be 4), the classical CRP formulation would use a constant $\mathbf{b} = 1$. Figure 5.5 presents the results using the classical formulation, where plate (a) shows the performance only after applying the CRP step, plate (b) shows the effect of stickiness adjustment and plate (c) provides the final result after the off-line revision step (using $\eta = \frac{1}{2K}$ given K classes from the online part, i.e., after stickiness adjustment). While the CRP step enables the framework to detect changes in time series characteristics, evidently, the stickiness adjustment is critical to control the 'hunting' behavior that creates 'too many' new classes. Finally, the off-line PFSA revision step helps to improve the result even further. It should be noted that this result is obtained using a noiseless (i.e., signal to noise ratio, SNR $= \infty$) data set with two features. Figure 5.6 presents the effect of noise content (for SNR $= \infty$, SNR $= 9$ and SNR $= 1$) on the performance using the same algorithm (i.e., constant $\mathbf{b} = 1$). The results demonstrate visually that the algorithm is quite robust to significant noise contamination.

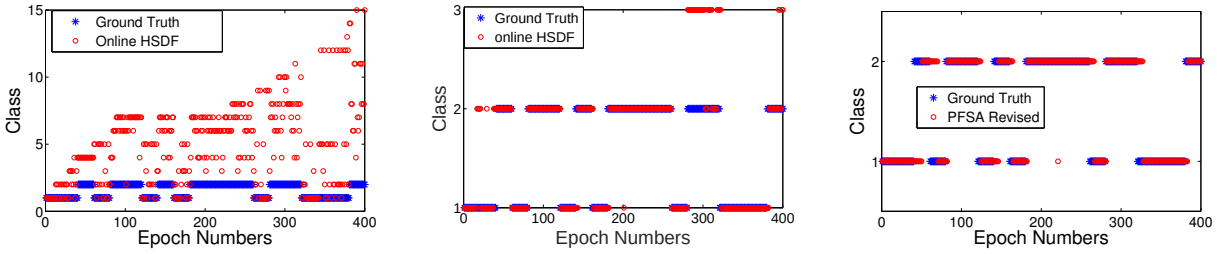
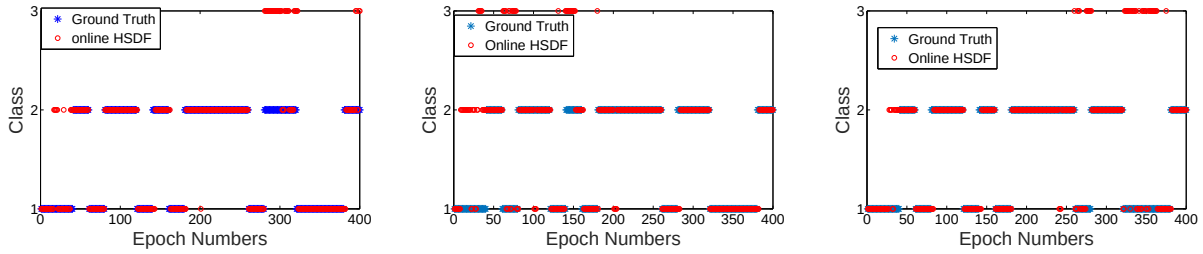


Figure 5.5: Results for noiseless Duffing system where, Plate (a) HSDF using classical CRP-only, Plate (b) HSDF using classical CRP and stickiness, and Plate (c) offline PFSA revision.

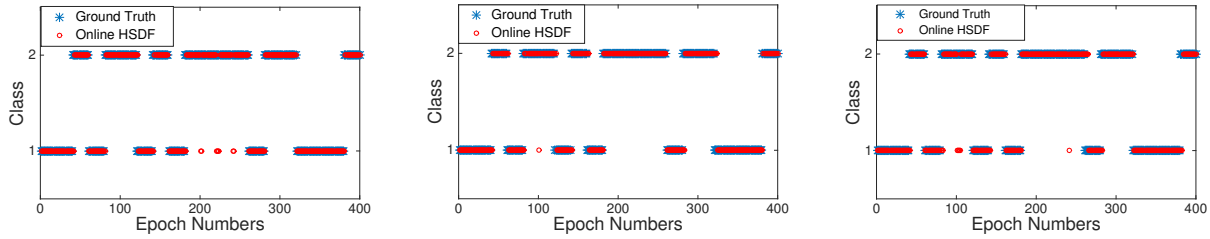
Performance with adaptive CRP formulation: Now we move to the adaptive CRP formulation as described in Algorithm 1, with an appropriate choice of $\mathbf{b}$ (i.e., equal to 1 or 2) based on the change in data likelihood of the existing classes. Results are presented in Figure 5.7 that show the adaptive formulation (with stickiness adjustment) to be quite efficient and achieves online performance better than that obtained after off-line revision with the classical formulation. Similar to the previous case, the algorithm is also quite stable under noise contamination. Additionally, we found that the sensitivity of the algorithm to the hyper-parameters, ϵ and κ reduces significantly



(a) (b) (c)

Figure 5.6: Results for noiseless Duffing system using on-line HSDF (i.e., without off-line PFSA revision) with classical CRP formulation under different noise contamination levels - Plate (a) SNR $= \infty$, Plate (b) SNR $= 9$ and Plate (c) SNR $= 1$.

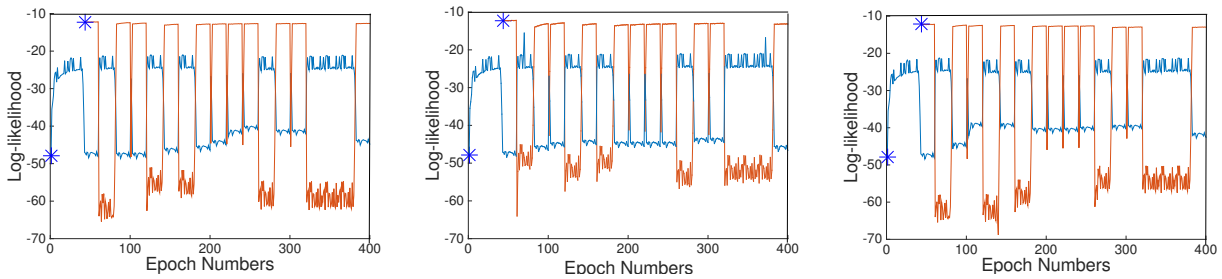
under the adaptive CRP formulation. Typically, the values for ϵ and κ used in the research are ≈ 0.02 and ≈ 0.6 respectively. Figure 5.8 shows the data log-likelihood plots for the class transitions and new class creation in an explicit manner. Table 5.1 compiles all the quantitative results for both classical and adaptive CRP formulation under the different noise conditions considered here. The results show that online HSDF with adaptive CRP performs the best under low to moderate noise level. At a higher noise level, the off-line revision may be more suitable.



(a) (b) (c)

Figure 5.7: Results for Duffing system using online HSDF (i.e., without off-line PFSA revision) with adaptive CRP formulation under different noise contamination levels - Plate (a) SNR $= \infty$, Plate (b) SNR $= 9$ and Plate (c) SNR $= 1$.

Performance comparison: We compared the results from our proposed algorithm with those from Hierarchical Dirichlet Process – Hidden Markov Model (HDP-HMM) (Fox et al. (2011)) that is a sampling based technique based on the Bayesian nonparametric concept, such that the joint



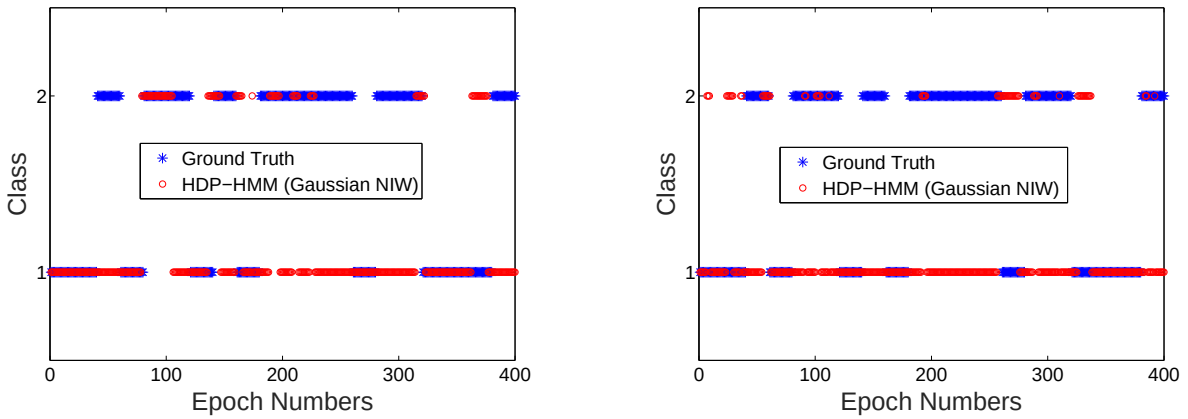
(a) (b) (c)

Figure 5.8: Data log-likelihood plots with $*$ representing the start of a new class for Duffing system using online HSDF (i.e., without off-line PFSA revision) with adaptive CRP formulation under different noise contamination levels - Plate (a) SNR = ∞, Plate (a) SNR = 9 and Plate (b) SNR = 1.

Table 5.1: Performance comparison of algorithm versions under different noise levels.

	Error %		
Algorithm	SNR = ∞ (∞ dB)	SNR = 9 (19.08 dB)	SNR = 1 (0 dB)
HSDF + Classical CRP	12.94	16.50	16.50
HSDF + Classical CRP + PFSA revision	5.50	6.25	**6.25**
HSDF + Adaptive CRP	**4.75**	**5.75**	7.00

distribution of the states are derived from the Dirichlet process. HDP-HMM techniques have been used in learning switching linear dynamical systems (SLDS). Note that the idea of stickiness has been adopted from the HDP-HMM literature as we aim to extract features for more realistic and general cases of nonlinear dynamical systems in a fast and computationally efficient manner.

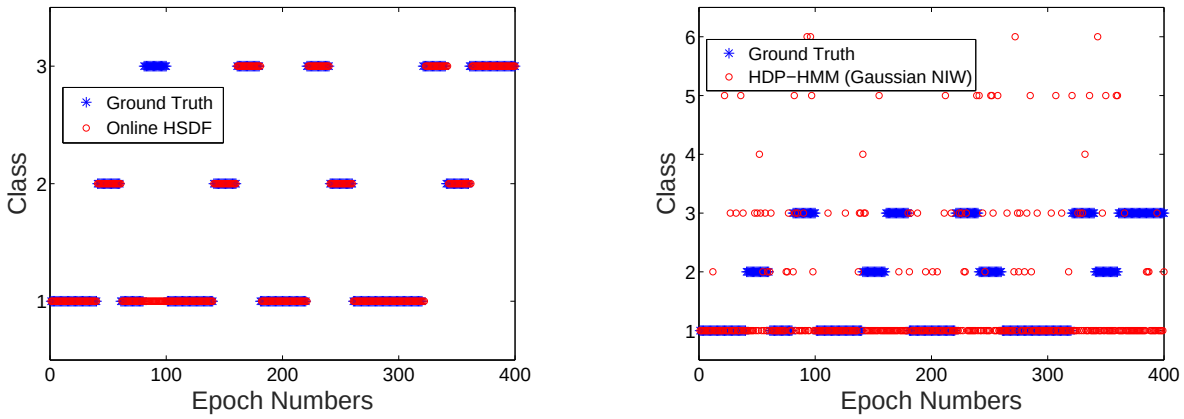


(a) (b)

Figure 5.9: Performance of HDP-HMM approach for time series with two features under different noise contamination levels - Plate (a) SNR = ∞, and Plate (b) SNR = 1.

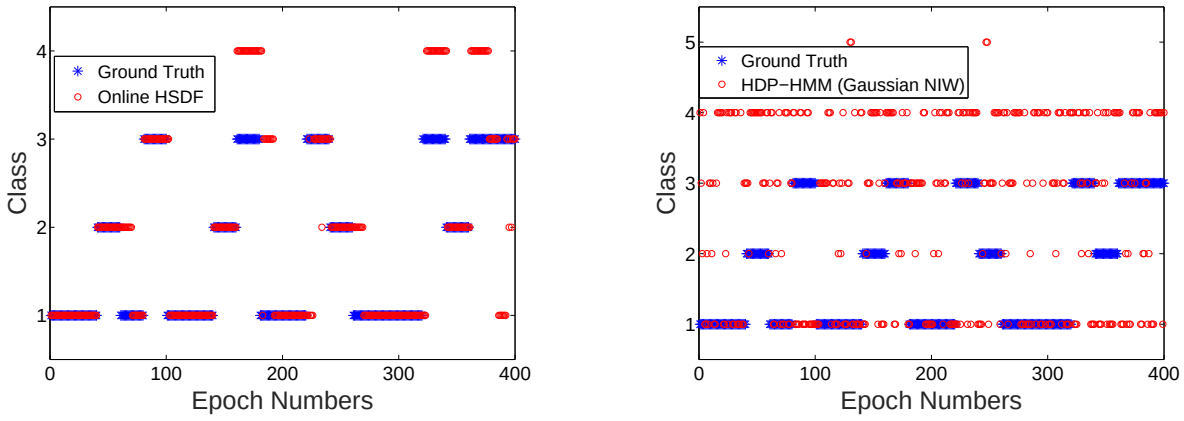
For comparison purposes, we implemented the codes made available on the authors' webpage [1] that uses a Gaussian observed model type with Normal-Inverse-Wishart (NIW) prior, which we found to be producing best results for all use cases. The results for the HDP-HMM approach under the noiseless (SNR = ∞) and the most noisy (SNR = 1) cases are shown in Figure 5.9. Note that the HDP-HMM Fox et al. (2011) algorithm classifies each data point individually, being a sampling technique. Hence, a majority voting was done to select the most prominent class in each epoch (i.e., 1000 data points as defined earlier) for a more realistic comparison. From the plots, it is quite evident visually that our proposed approach has better accuracy which can be explained by the fact that SDF inherently is an efficient way to model nonlinear system behavior. Also, the performance of the HDP-HMM approach suffers significantly in the presence of a large amount of noise in the signal. However, we note that HDP-HMM approach correctly identifies the number of

[1] https://www.stat.washington.edu/ ebfox/software.html



(a) (b)

Figure 5.10: Performance comparison of (a) HSDF and (b) HDP-HMM approaches for time series with three features under no noise condition.



(a) (b)

Figure 5.11: Performance comparison of (a) HSDF and (b) HDP-HMM approaches for time series with three features under SNR = 1 condition.

features present in the non-stationary time series data which is significant. We performed further comparison using the 3-features case, and present the results in Figure 5.10 and Figure 5.11 for the SNR $= \infty$ and SNR $= 1$ cases respectively. It can be observed that our algorithm still performs better in both the identification of the total number of features present in the time series data as well as in classifying them. Finally, the quantitative performance of HSDF and HDP-HMM (Gaussian NIW option) for the two and three features cases are summarized in Table 5.2. Note, the smaller time requirement for our proposed algorithm compared to that of HDP-HMM is primarily due to the fact that there is no sampling step or latent variable involved in the HSDF approach. The computation time reported here is achieved with MATLAB implementations on a 2.1 GHz Intel Xeon(R), 1200 MHz CPU with 64GB RAM and UNIX OS.

Table 5.2: Performance comparison of HSDF and HDP-HMM approaches on simulated datasets

Method		**Online HSDF**		**HDP–HMM (Gaussian NIW)**	
Noise level (dB)		SNR $= \infty$	SNR $= 0$	SNR $= \infty$	SNR $= 0$
2 features	Error(%)	4.75	7.00	48.25	57.50
	Time($secs$)	66.1	65.0	423.0	419.7
3 features	Error(%)	7.25	26.50	56.25	76.75
	Time($secs$)	88.1	112.4	418.0	426.8

5.5.3 Real use case of energy disaggregation

The aim of energy diaggregation is to determine the energy (typically electrical) consumed by each end use (or appliance) in a building at multiple time periods given the total (mains or transformer) power consumption signal. Typically, supervised approaches are used to solve the disaggregation tasks where unique end use signal characteristics are learned using labeled data and the inference process aims to detect such characteristics from the total power consumption signal. However, labeled data for every possible end uses (and their complex combinations) may be extremely difficult to obtain in real life. Therefore, unsupervised approaches that can reliably detect unique signal characteristics and the transitions from one to another can be extremely useful in

deploying energy disaggregation algorithms in the field. Therefore, our proposed algorithm can be an excellent candidate for this purpose. To demonstrate the efficacies of our approach in this regard, we leverage the publicly available Reference Energy Disaggregation Dataset (REDD) (Kolter and Johnson (2011)). This data was collected from 6 homes in Boston Massachusetts over a period of 3-19 days. The dataset has two varieties: low frequency and the high frequency. The high frequency versions are taken at 15KHz, thus requiring high precision and costlier hardware setup compared to the low frequency setup. From the standpoint of hardware and storage cost reduction as well as research with a wider impact, we implement our online HSDF scheme on the low frequency dataset. The low frequency dataset contains two mains power that were recorded at 1Hz frequencies, while the end use consumption were recorded at 2/7Hz giving a total data record $321,984$ data points (see Batra et al. (2014); Kolter and Johnson (2011) for a more detailed description of REDD).

5.5.4 Ground truth formulation, results and performance comparison

In order to establish certain ground truths for the study involving REDD, signals from individual end uses at different time instances were mixed-and-matched, similar to how the simulated signals were generated, but with the distinction that this are recorded real-life datasets. Such mix-and-match was required to ensure reliable ground truth for our study as well as circumvent certain issues due to the non-uniformity in the sampling frequency of the mains power and the end uses. For our test case, individual end use and their complex combinations are identified by simply giving class labels, i.e., 1, 2, $\cdots$, The two test cases considered are described as follows:

Homogenous features case study: In this case, an end use or a combination of end uses was described by a single time series pattern selected from the REDD. Note, due to variations in usage patterns and other load requirements, a particular end use or a certain combination of end uses may demonstrate different signal characteristics at different time periods. Such scenario was termed 'energy disaggregation with heterogeneous features' (investigated in the next case study), however this case study deals with homogeneous features where test time series were generated by randomly concatenating unique time series segments representing different classes, i.e., different

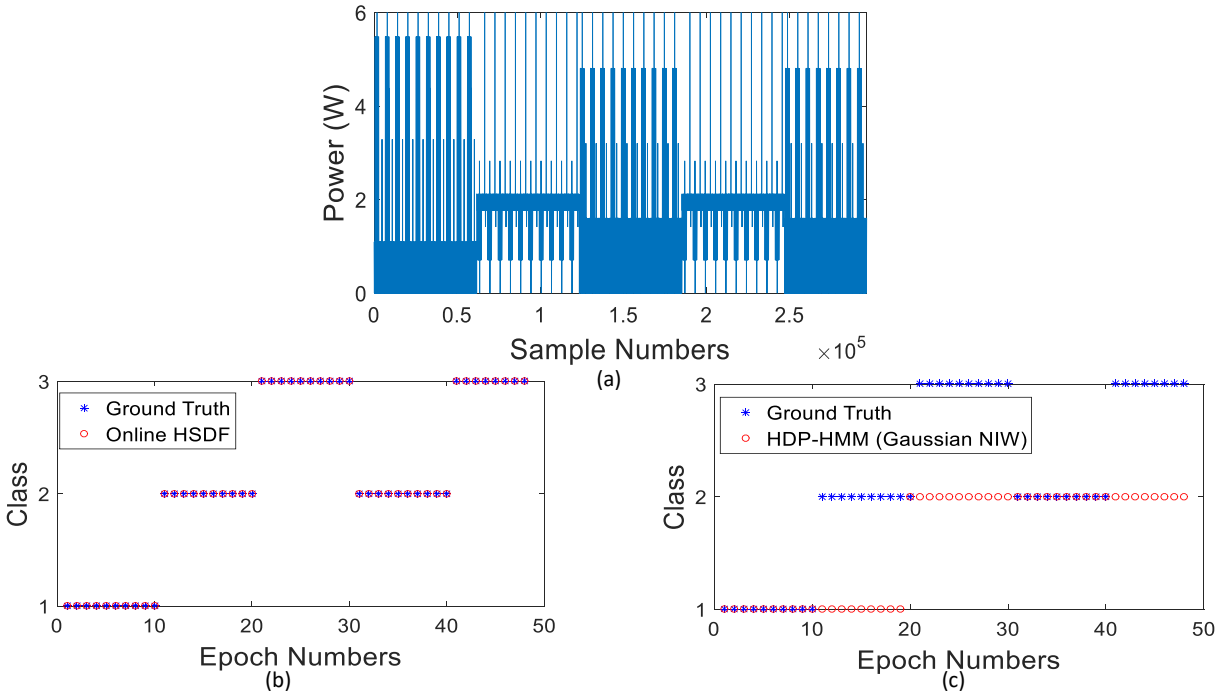


Figure 5.12: Concatenated power consumption for 3 randomly chosen end uses with homogenous characteristics in (a), and corresponding feature identification results via: (b) Online HSDF and (c) HDP-HMM; at slow time epochs each consisting of $6,172$ fast scale datapoints ≈ 6 hours.

end uses and their combinations. This case study is particularly useful to test the robustness of the proposed algorithm in the sense that it should be able to identify a previously observed pattern correctly and not learn new PFSA model to represent it. Again, the associated ground truth is available since the test dataset was constructed with only some of the end uses of REDD. Figures 5.12(a) and 5.13(a) show the time series for $K = 3$ (with slow time epochs each consisting of $6,172$ datapoints at the fast time scale ≈ 6 hours) and 4 (with slow time epochs each consisting of $12,384$ datapoints at the fast time scale ≈ 12 hours) classes respectively that results from concatenating certain fixed patterns from different end uses and from their complex combinations. The results in Figures 5.12(b) and 5.13(b) show that algorithm performs accurately (0% error) when the features are homogeneous. Note that in these results, online HSDF is able to effectively perform both class retention and transitions to a new or previous observed class. When compared to the performance of HDP-HMM results shown in Figures 5.12(c) and 5.13(c) (with errors of 56% and 60% respectively.

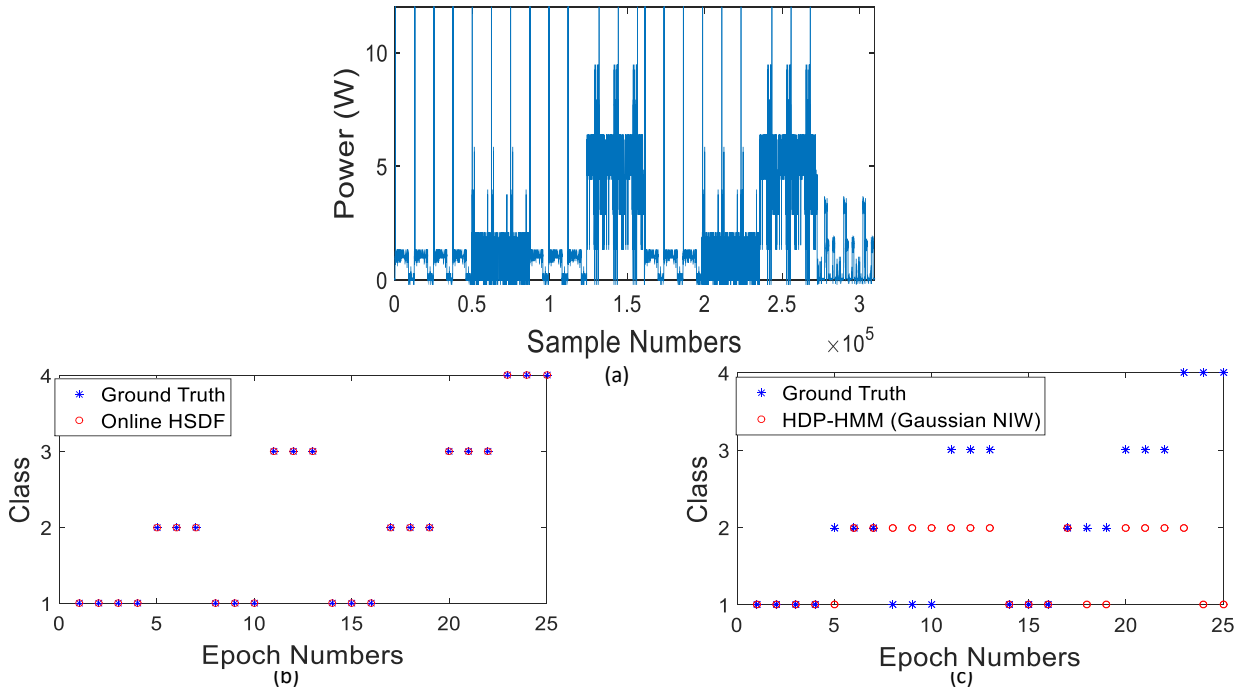


Figure 5.13: Concatenated power consumption for 4 randomly chosen end uses with homogenous characteristics in (a), and corresponding feature identification results via: (b) Online HSDF and (c) HDP-HMM; at slow time epochs each consisting of $12,384$ datapoints $\approx$ 12-hours.

the wide error margin can be possibly attributed to excessive stickiness). Online HSDF is however able to overcome such problems via using the rate of change of data likelihood along with the actual values of likelihood with respect to the existing classes.

Heterogenous features: Here, a particular end use is (or a certain combination of end uses are) not constrained to have a single pattern at different time periods. As for a certain end use (or combination), patterns were randomly selected from different time periods such that there is a high chance that time series patterns may be quite different for the same ground truth label. Therefore, the unsupervised algorithms can generate new classes for the same ground truth label. However, algorithm was evaluated using this case study to investigate if it is able to identify certain underlying similarities between seemingly different patterns with the same ground truth label. Figures 5.14 and 5.16 show the time series used in this case study with $K = 3$ and 4 (ground truth) classes respectively. Figures 5.15(a) and 5.17(a) show the performance of the online HSDF

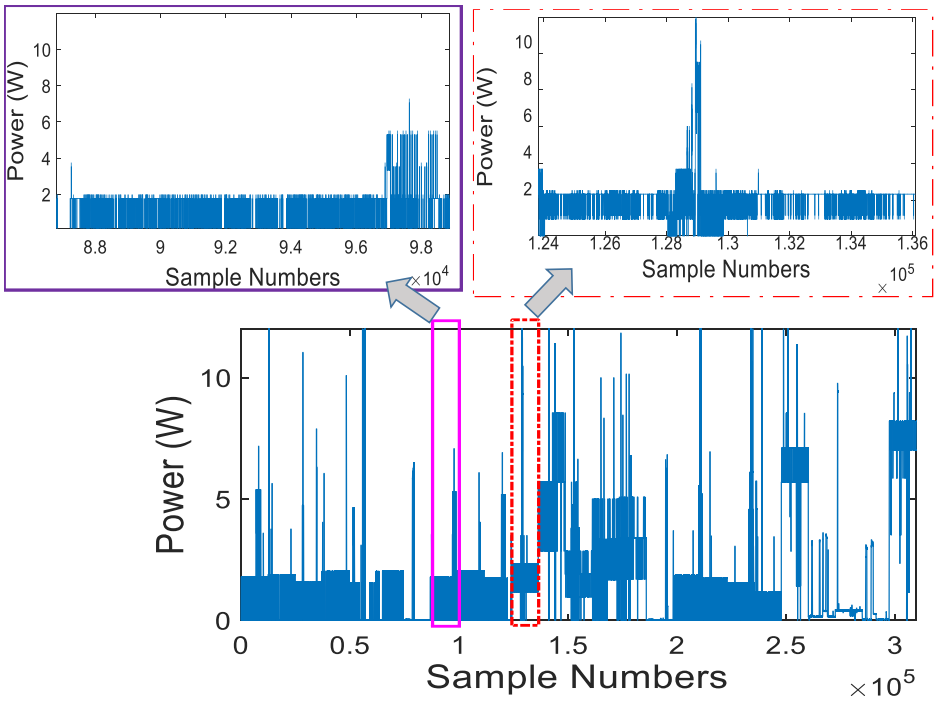
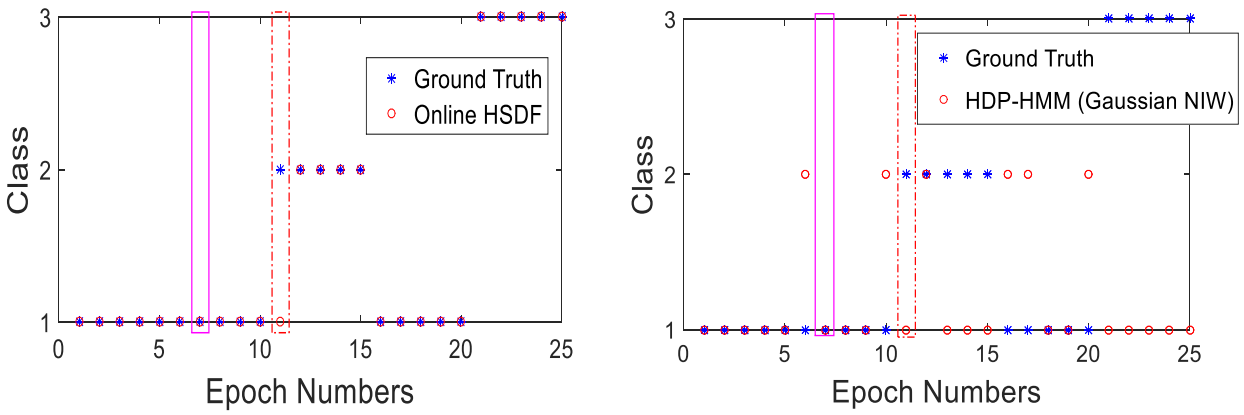


Figure 5.14: Concatenated power consumption for 3 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features with different ground truth labels).

algorithm and Figures 5.15(b) and 5.17(b) show the performance of the HDP-HMM algorithm for the time series with 3 and 4 (ground truth) classes respectively. While the overall performance deteriorates compared to that in the homogeneous case study for both algorithms, online HSDF still performs better than the HDP-HMM algorithm. Further analyzing the errors, a few regions - purple (solid) and red (dash-dot) boxes marked in Figure 5.14 represent two different features but identified as similar by both algorithms due to their mostly similar characteristics. On the other hand, similar highlights in Figure 5.16 show two slow time-scale epochs with the same ground truth labels. Both algorithms fail to detect the similarities due to significant difference in scale and other characteristics. The slow time scale epochs highlighted in Figures 5.14 and 5.16 are also shown by similar purple (solid) and red (dash-dot) boxes in Figures 5.15 and 5.17 respectively. To address such issues, a more suitable (possibly optimized) slow time-scale epoch length may need to be identified to capture the quasi-stationary characteristic which will be an important future work.



(a) (b)

Figure 5.15: Feature identification results of concatenated power consumption for 3 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features misclassified by both algorithms as similar while they have different ground truth labels) via: (a) Online HSDF and (b) HDP-HMM; at slow time epochs of $\approx$ 12 hours.

Remark 5.5.1. The best results obtained for the HDP-HMM required the use of stickiness options in all the cases considered, otherwise many unstable transitions occur in a hunting like situation. On the other hand, with 'stickiness', it tends to be overly sticky leading to some errors as seen in the results. We believe that this observation is most likely due to the choice (in the publicly available code we used) HDP-HMM's stickiness hyper-parameter having only 2 states – either sticky or not unlike online HSDF's adaptive stickiness capability.

Table 5.3 summarizes the results for comparing online HSDF with HDP-HMM. Note, the 3 class scenario in the homogeneous features case study requires relatively longer time as it uses a reduced epoch length, resulting in more slow time-scale epochs (50 as opposed to 25 in other cases) for computation as seen in Figure 5.12.

For emphasis, all the computation times reported here are based on the MatLab implementations of the algorithms.

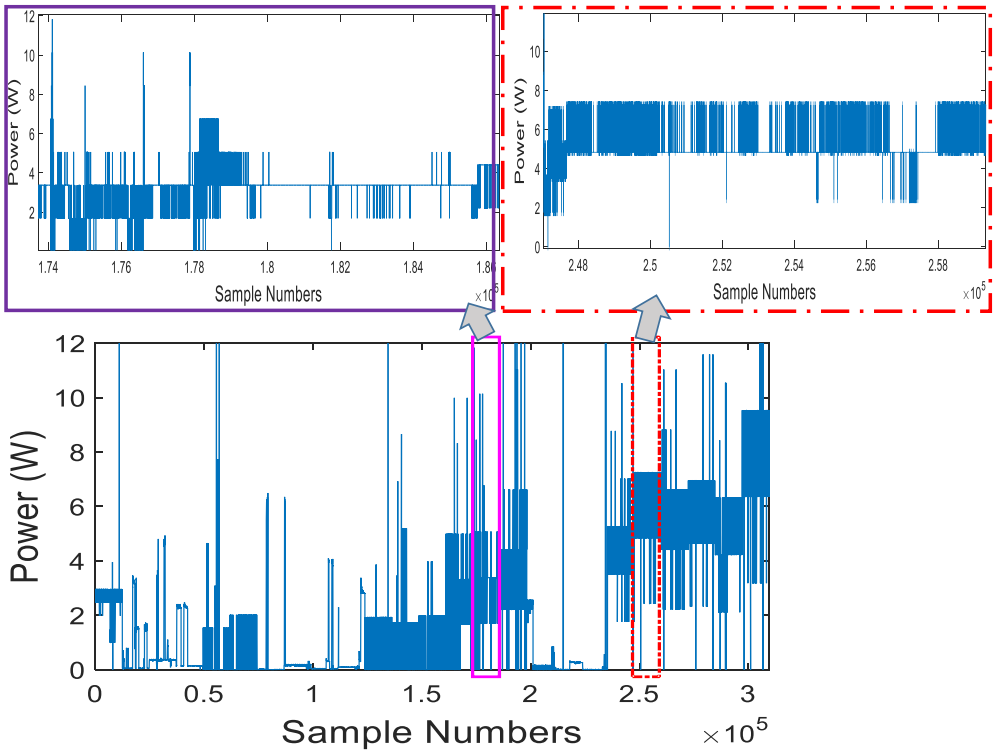
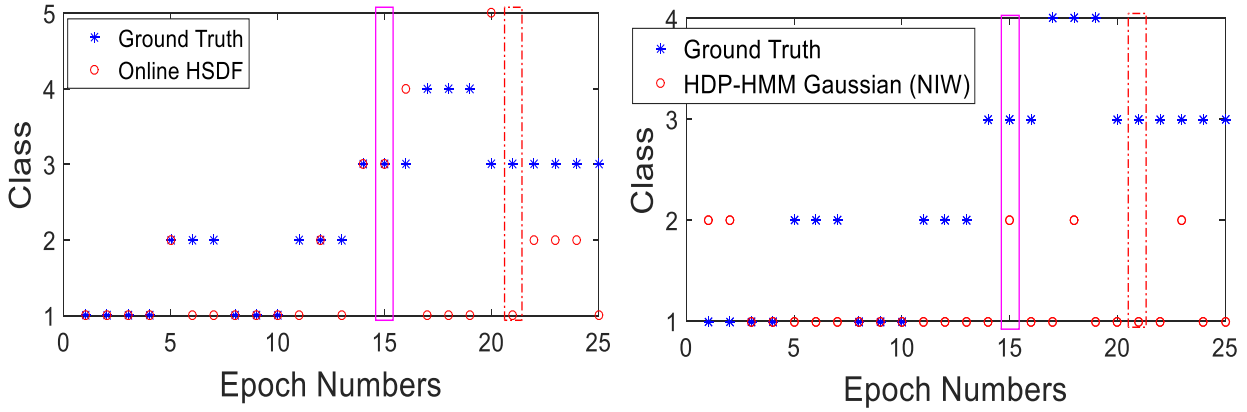


Figure 5.16: Concatenated power consumption for 4 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features with same ground truth label 3).

5.6 Summary

To summarize, a hierarchical feature extraction technique has been formulated to extract and analyze features from time series dataset of potentially increasing (streaming type) time records. The algorithm showed that the rate of change of likelihood was more beneficial for reducing model complexity than the known likelihood when learning models from dataset. The proposed algorithm is tested and validated using time series data generated from well-known nonlinear dynamical system simulation involving Duffing and Van der Pol equations as well as a publicly available reference energy disaggregation dataset (REDD) related to the important problem of energy disaggregation. We demonstrate the efficacy of the algorithm under various noise contamination levels and in comparison with the competing HDP-HMM approach. We note that a key advantage of the proposed



(a) (b)

Figure 5.17: Feature identification results of concatenated power consumption for 4 randomly chosen end uses with heterogenous characteristics (parts highlighted in the solid purple and red dash-dot boxes represent features misclassified by both algorithms as different while they have the same ground truth label 3) via: (a) Online HSDF and (b) HDP-HMM; at slow time epoch of ≈ 12 hours.

Table 5.3: Performance comparison of HSDF and HDP-HMM approaches on low frequency REDD

Method		**Online HSDF**		**HDP-HMM (Gaussian NIW)**	
Cases		3 classes	4 classes	3 classes	4 classes
Homogenous features	Error(%)	**0.00**	**0.00**	56.00	60.00
	Time(*secs*)	5.9	8.3	1818.3 †	447.2 †
Heterogenous features	Error(%)	**4.00**	56.00 ‡	56.25 ‡	80.00 ‡
	Time(*secs*)	3.8	4.3	598.9 †	454.6 †

† The reported time may not be optimum since Thomas Minka's lightspeed toolbox support Minka (2017) was not compatible with the current MATLAB 2017 version. Also the algorithm requires an unspecified number of Gibbs sampling.

‡ Indicates very large errors partially due to ground truth issues as well as failure scenarios for the algorithms.

technique is its low computational and memory complexity. Hence, it can be extremely suitable for on-board real time applications.

CHAPTER 6. FEATURE EXTRACTION FROM SPATIOTEMPORAL DATA – SPATIOTEMPORAL PATTERN NETWORK (STPN)

In this chapter, a spatiotemporal pattern network (STPN) approach is proposed for analyzing energy demand by individual household appliances from the whole building dataset and predicting the energy supplied by an array of wind turbines. The ideas behind the approach are methodically and mathematically presented, and the algorithm is subsequently tested on real-life spatiotemporal observations.

6.1 Introduction

Symbolic Dynamic Filtering (SDF) have been described in Subsection 2.7.2, and an application of it shown in Chapter 5. Furthermore, in Subsection 2.7.3 a D-Markov extension of SDF was described. Here, multiple applications are proposed: one furthers the application of SDF for temporal applications while the other includes spatial consideration of the temporal dynamic systems.

To facilitate energy prediction for systems with both spatial and temporal characteristics, probabilistic graphical models (PGMs) may be employed as the spatiotemporal interactions are naturally suited for graph representation and can be evaluated by the associated probabilities. PGM encompasses a variety of models described by conditional dependence structures, so-called graphs, including Bayesian networks and undirected/directed Markov networks, and can be used to deal with dynamical systems and relational data (Koller and Friedman (2009)). Bayesian networks are a type of PGM that capture causal relationships using directed edges (Koller and Friedman (2009)), where the overall joint probability distribution of the network nodes (variables) is computed as a product of the conditional distributions (factors) defined by the nodes in these network. However, prediction problems are not straightforward for Bayesian networks, as they only encode node-based conditional probabilities, and the approximation of the joint distribution using node-based struc-

tures is often intractable. This is because a certain directed acyclic graphical structure may not allow for easy and exact computation of certain probabilities related to inference questions.

6.2 Background and Motivation

To address prediction problems in energy systems, this chapter presented a new data-driven framework (namely spatiotemporal pattern networks, or STPN) to leverage the spatiotemporal interactions of energy systems for prediction. STPN was built on SDF to capture the spatiotemporal characteristics of complex energy systems. It also implements prediction at both spatial and temporal resolutions. For validation purpose, the proposed approach was evaluated on two representative case studies. The first was taken from the energy supply side, wind power prediction in a large-scale wind farm. The second case study was from the energy demand side, home energy disaggregation (also as non-intrusive load monitoring (NILM), a well-established problem that involves disaggregating the total electrical energy consumption of a building into its constituent load components without the necessity for extensive metering installations on individual household or appliances (Hart (1992); Zeifman and Roth (2011); Cominola et al. (2017)).

The main reason for choosing an energy production system and the non-intrusive load monitoring problem on the demand side, was to demonstrate the effectiveness of the proposed approaches on both sides of the energy meter. Note that as penetration of renewable energy systems increases, prediction accuracy becomes ever more important. This is because without accurate prediction of renewable energy production, it is difficult to control the power distribution, pricing and scheduling of other energy sources. Insights into the electric load breakdown were required in order to perform effective demand response and load shaping for peak power reduction. Furthermore, if inexpensive energy disaggregation approaches are widely deployed, actionable spatiotemporal information could be obtained on the types of load components that could respond to local overproduction of renewable energy such as wind power.

The main contribution was the demonstration that the proposed data-driven modeling scheme could efficiently learn spatiotemporal characteristics of a distributed energy system in a scalable

and computationally efficient manner. The modeling scheme can enable high-accuracy prediction of energy production (for a distributed generation system such as wind farm) and energy consumption (for a complex combination of electrical energy end uses in a building). For wind turbine power prediction, the spatiotemporal characteristics between different wind turbines are identified, while for home energy disaggregation the complex coupled temporal features are revealed. An STPN-based convex programming method is presented in this work in order to improve energy prediction and disaggregation performance. Algorithm's performance was compared with other competitive and state-of-the-art data-driven modeling techniques to demonstrate the significant improvement in accuracy recorded. While energy prediction is critical, the data-driven modeling strategy also opens up many other applications such as performance monitoring, fault diagnostics, control, and optimization in many large energy systems that are difficult to model using traditional physics-based principles.

6.3 Spatiotemporal Pattern Network

In this section we presented the construction details of spatiotemporal patter network (STPN) for dynamical systems, A and B, based on the concepts of SDF (Subsection 2.7.2). Such formulation is usually preceded by discretization (Subsection 2.7.1).

6.3.1 Symbolic modeling of dynamical systems and interactions

Suppose there are two different dynamical systems A and B. In real-world problems, such as wind power prediction, A and B can represent two different wind turbines in a large wind power farm. Alternatively, in residential home energy disaggregation, A and B could represent HVAC system electricity consumption and appliance electricity consumption. For each system, there are several measured variables and typically some key observations are selected to establish the model for analysis. For example, in a wind turbine, wind speed and wind power are the two key observations for power prediction. It is noted that some other variables, such as wind direction and the rate of its change, possibly affect power such that these variables can also be taken into

account. As described in Subsection 2.7.1, there are numerous approaches that can be used; In the research, maximally bijective discretization (MBD) was applied to the supply side dynamical systems (wind turbines) and maximum entropy partitioning was used in the demand side dynamical systems (HVAC, appliances, etc.). Wind speed and wind power are chosen and their input-output relation in the continuous domain can be maximally maintained using the maximally bijective discretization. However, for residential home energy disaggregation, the unique variable for each part of home energy use is the energy consumption itself such that there is no input-output relation in the continuous domain.

Figure 2.13 showed the symbol sequence generation in the form of PFSA using two different methods, i.e., maximally bijective discretization and maximum entropy partitioning, respectively. As described in Subsection 2.7.3, a D-Markov machine can be represented by a PFSA using previous D symbols to indicate a particular state. In Subsection 2.7.3, two different systems to quantify their spatiotemporal or temporal relationships are considered. From Figure 6.1, the state transition matrices Π^A and Π^B show the self-relations of systems A and B respectively. Then the cross-state transition matrices represented here by Π^{AB} and Π^{BA}, correspondingly, represent the cause-effect relations from A to B and B to A, respectively. However, it should be noted that such causal dependencies between systems A and B are not necessarily equivalent. To quantify the relationships in a D-Markov machine, a xD-Markov machine, atomic patterns (AP) and relational patterns (RP) were introduced in Sarkar et al. (2014), which can offer more detail. More formally, the entries of the cross-state transition matrices Π^{AB} and Π^{BA} can be expressed by:

$$\pi_{k\ell}^{AB} := P\left(s_{n+1}^{B} = \ell \mid s_{n}^{A} = k\right) \ \forall n$$

$$\pi_{ij}^{BA} := P\left(s_{n+1}^{A} = j \mid s_{n}^{B} = i\right) \ \forall n$$

where $j, k \in Q^A$ and $i, \ell \in Q^B$ for states set Q satisfying the property in Subsection 2.7.2. The above relations show that a cross-state transition matrix can be constructed from symbol sequences obtained from two different dynamical systems while every entry of each matrix signifies the tran-

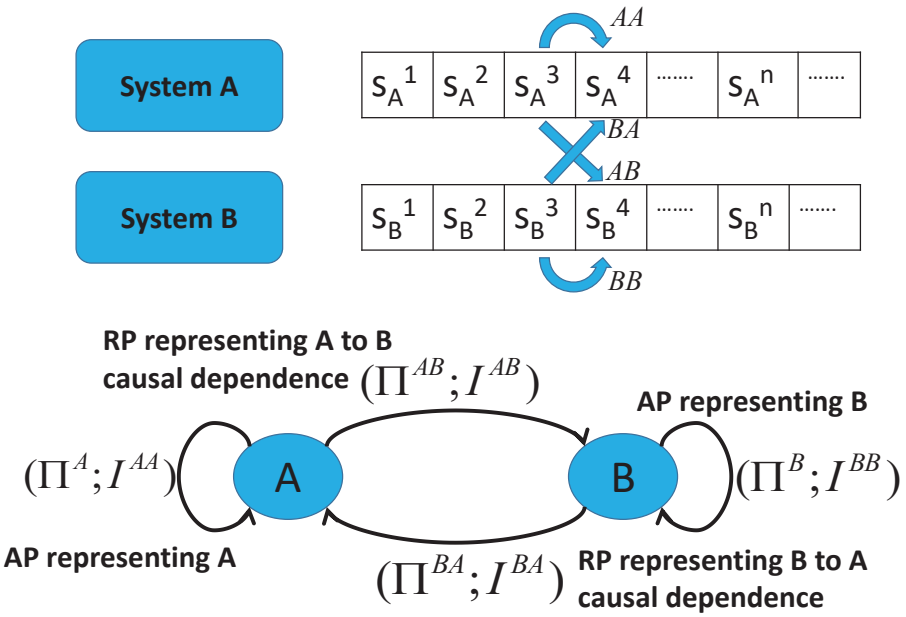


Figure 6.1: Construction of STPN: Atomic patterns (APs) and relational patterns (RPs) formulation where s represents the symbol.

sition probability from one state in the first dynamical system to another state in the second dynamical system. For instance, π_{ij}^{BA} means the transition probability from state i in the system B to another state j in the system A.

Moreover, we use an information-theoretic metric in order to quantify the value of the atomic and relational patterns (the research emphasized more on the relational patterns). In this context, mutual information is a metric of interest introduced to address the quantification. For example, from Figure 6.1, we denote by I^{AA} and I^{AB} the mutual information of the atomic and relational patterns, respectively, associated with systems A and B. Formally, the atomic pattern of system A is expressed as follows:

$$I^{AA} = I(s_{n+1}^A; s_n^A) = \mathcal{H}(s_{n+1}^A) - \mathcal{H}(s_{n+1}^A | s_n^A)$$

where

$$\mathcal{H}(s_{n+1}^A) = -\sum_{i=1}^{\mathcal{Q}_A} P(s_{n+1}^A = i) \log_2 P(s_{n+1}^A = i)$$

$$\mathcal{H}(s_{n+1}^A|s_n^A) = -\sum_{i=1}^{\mathcal{Q}_A} P(s_n^A = i)\mathcal{H}(s_{n+1}^A|s_n^A = i)$$

$$\mathcal{H}(s_{n+1}^A|s_n^A = i) = -\sum_{i=1}^{\mathcal{Q}_A} P(s_{n+1}^A = l|s_n^A = i)\cdot$$
$$\log_2 P(s_{n+1}^A = l|s_n^A = i)$$

Therefore, based on the quantity I^{AA} (defined using entropy $\mathcal{H}$ values as presented above), the temporal self-prediction capability of system A can be identified.

On the other hand, the mutual information for the relational pattern involved in systems A and B can be described as:

$$I^{AB} = I(s_{n+1}^B; s_n^A) = \mathcal{H}(s_{n+1}^B) - \mathcal{H}(s_{n+1}^B|s_n^A)$$

where

$$\mathcal{H}(s_{n+1}^B|s_n^A) = -\sum_{i=1}^{\mathcal{Q}_A} P(s_n^A = i)\mathcal{H}(s_{n+1}^B|s_n^A = i)$$

$$\mathcal{H}(s_{n+1}^B|s_n^A = i) = -\sum_{i=1}^{\mathcal{Q}_B} P(s_{n+1}^B = l|s_n^A = i)\cdot$$
$$\log_2 P(s_{n+1}^B = l|s_n^A = i)$$

Hence, the quantity of I^{AB} identifies system A's capability of predicting system B's outputs and vice versa for I^{BA}. Furthermore, based on mutual information, patterns can be assigned with weights such that some patterns with low mutual information may be rejected in order to simplify the model (Sarkar et al. (2014)). Based on the above analysis, it has been shown that STPN can be an effective tool for capturing the spatiotemporal interactions between different dynamical systems. To validate such a data-driven method, two case studies are presented in terms of supply side dynamical systems (i.e., wind turbines in a wind farm) and demand side dynamical systems (i.e., home electric energy disaggregation), which demonstrate the efficacy and effectiveness of STPN. The prediction process can be described as follows: Given a training data set in the continuous

domain, we use partitioning methods to discretize and symbolize the data for running the xD-Markov machine. The probability transition matrices are obtained for predictions in symbolic or continuous domains. For symbolic prediction, we find the most likely symbol sequence for system A given another symbol sequence of system B running the xD-Markov model numerous times. While in the continuous domain, the prediction can be acquired based on the symbolic prediction using expectation as follows:

$$W(k) = \sum_{j=1}^{m} Pr_k(j) W(E|j) \tag{6.1}$$

where, $W(k)$ represents the expectation of energy at the k^{th} instant, $Pr_k(j)$ signifies the probability of j^{th} symbol occurring at the k^{th} instant after running numerous simulations of Monte Carlo Markov Chain, $W(E|j)$ indicates the expectation of energy for the discrete bin labeled by symbol j (suppose that in that bin there are m discrete symbols). The pseudo-code of energy prediction based on STPN is as follows.

Input: Training data sets of systems i, C_i' (i represents any system), depth of D,

Output: Predicted results $\hat{C}_i$,

Discretize and symbolize the continuous data C_i' to s_i;

Calculate state transition matrices and mutual information for s_i;

Calculate the expected value of energy in the discrete bin;

Use Equation 6.1 to calculate the prediction results $\hat{C}_i$;

Algorithm 2: Energy prediction based on STPN.

6.4 Supply Side: Wind Turbines

6.4.1 Geographical information

In this subsection, a case study based on the supply side of energy systems, here wind turbines, is used to validate the data-driven method proposed in this work. The STPN framework is used in a wind turbine network in order to capture the causal dependencies among different turbines that can be regarded as sub-systems of a wind farm. The current study used the 2006 Western Wind

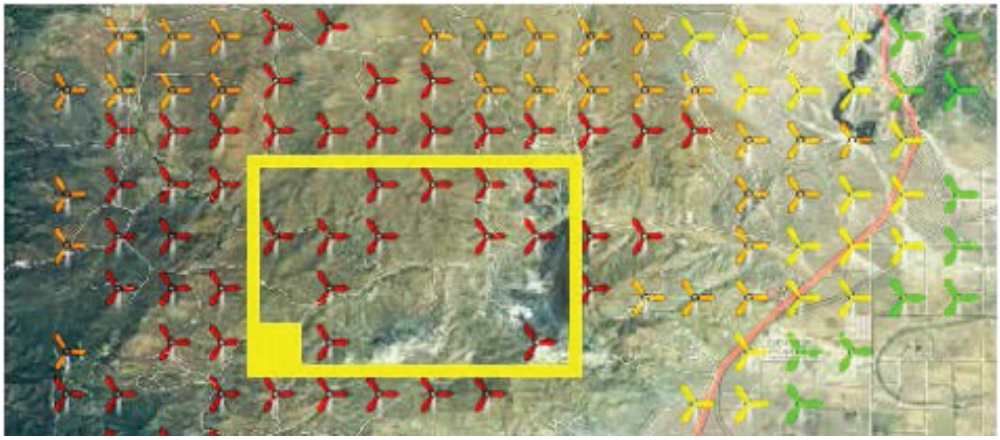


Figure 6.2: Geographical information of wind turbines under analysis which are located in California, between 35.28-35.33N and 118.09-118.17W.

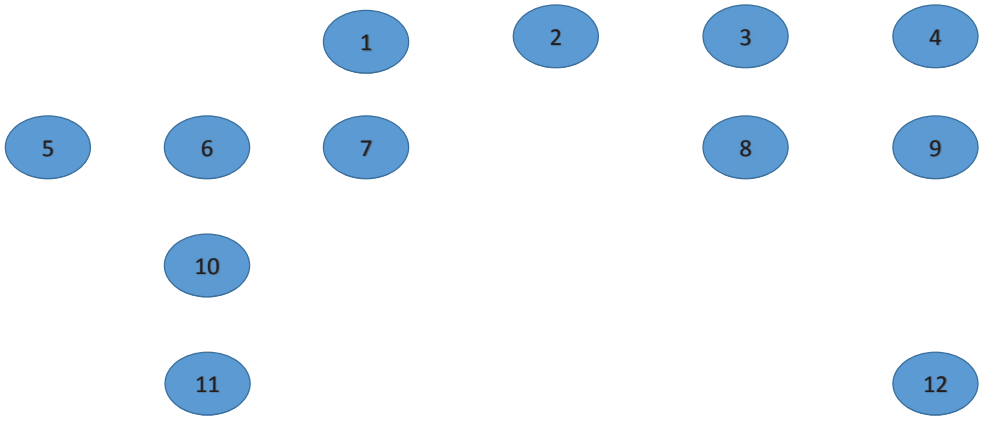


Figure 6.3: Representation of STPN for 12 wind turbines.

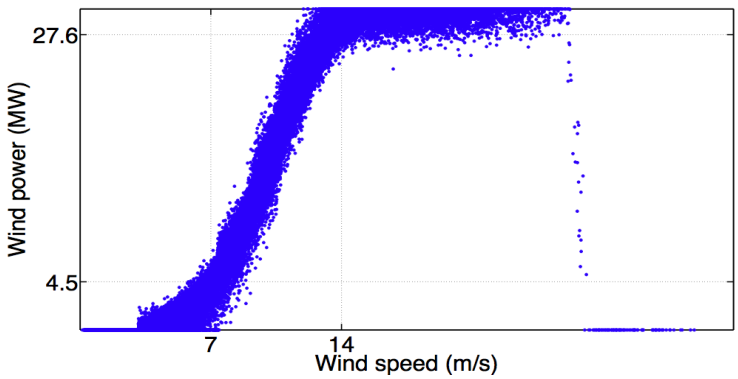


Figure 6.4: Discretization of a typical wind turbine systems using maximally bijective discretization (MBD).

Integration data set obtained from NREL (Laboratory (2017)) to uncover causal dependencies which are important for individual wind turbine power prediction in a mutual turbine-turbine setting. For establishing the STPN, twelve wind turbines (located in California) are chosen; their IDs can be identified as: 4494, 4495, 4496, 4497, 4423, 4424, 4425, 4426, 4427, 4361, 4313 and 4314 (labeled by 1-12) in this context, and the capacity factors are between 41% and 45% approximately. For completeness, the geographical information of the wind turbines is also provided. The annual average wind velocity in the area where the considered turbines are located is around 9 m/s, with an elevation from 1019 to 1207 m above mean sea level.

As shown in Figure 6.2, twelve wind turbines are distributed in various locations, which can be identified as nodes in the STPN represented by Figure 6.3. From Figure 6.4, the relationship between wind speed and wind power can be observed with other wind turbines exhibiting a similar pattern. The input-output relation involving a wind turbine is significant such that MBD enables the maximum preservation of their correspondence in the symbolic domain. Figure 6.5 shows an instance of symbol sequence for a wind turbine and it can be observed that most of the symbols are 1, 5, 8 and 9.

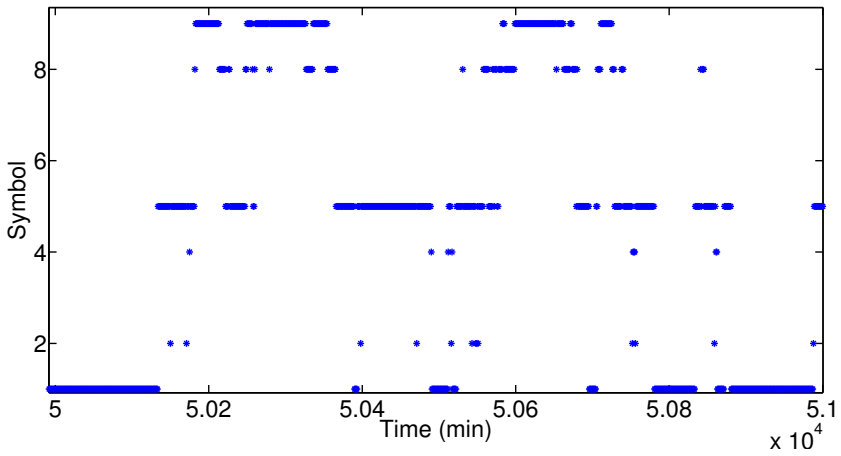


Figure 6.5: Symbol sequence plot for a typical wind turbine.

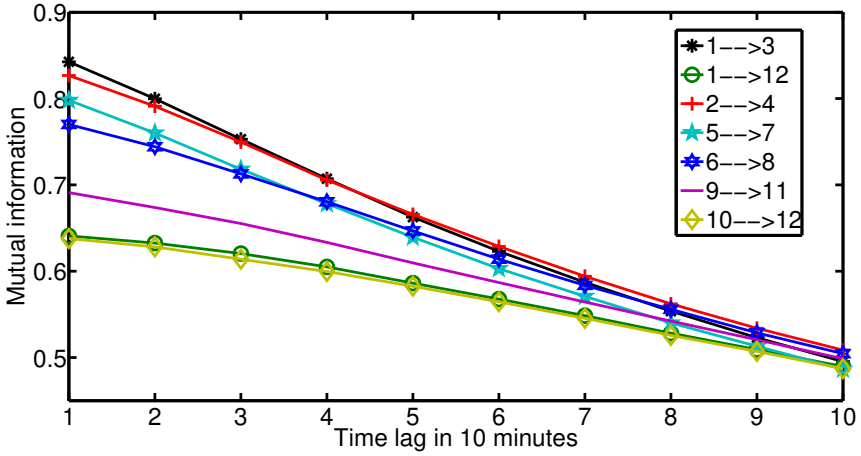


Figure 6.6: Mutual information of relational patterns for selected pairs of wind turbines

6.4.2 Results and discussion

The mutual information of RP between a pair of wind turbines will first be investigated according to the state transition matrices generated by xD-Markov machines. We set the depth as 1 for simplicity, though one can increase the memory parameter. This implies that the current state of one selected wind turbine depends only on the last state of another selected wind turbine. The effect of time lag on the mutual information between wind turbines is studied to address the temporal characteristics. The results in Figure 6.6 show that as the time lag increases, the mutual information decreases correspondingly. Thus, in this work one can maximize the causal dependencies between any two different wind turbines at time lag 1.

Remark 6.4.1. In the case study presented here, we have considered 12 wind turbines and hence, we have $12(12-1) = 132$ possible relational patterns. After removing the relational patterns with very low MI, we have shown the evolution of MI for 7 selected relational patterns in Figure 6.6 only to demonstrate the effect of time lag on their pair-wise dependencies. As the area is located in the state of California, the wind direction from West to East is dominant although various other wind directions are observed in the area during different time periods. While we observe high MI in a West to East direction, we would like to note that high MI essentially signifies the capability of predicting a certain turbine's energy production given the energy production of another turbine. Such predictability, while it should depend on the wind direction, may also depend on many other unmodeled dynamics and factors (that may be intractable by traditional physics-based methods), which are captured by the proposed modeling framework.

The spatial characteristics between two different wind turbines is another critical factor in STPN. Wind turbines labeled by 5, 6, 7, 1, and 10 are chosen for the purpose of such an analysis. Figure 6.7 shows that the causal dependency between any two wind turbines reduces with the increase of geographical (spatial) distance between them along any direction. Figure 6.8 illustrates that the metric based on mutual information for all pairs of wind turbines as a function of the Euclidean distance between them exhibits a generally decreasing trend. Consequently, in summary,

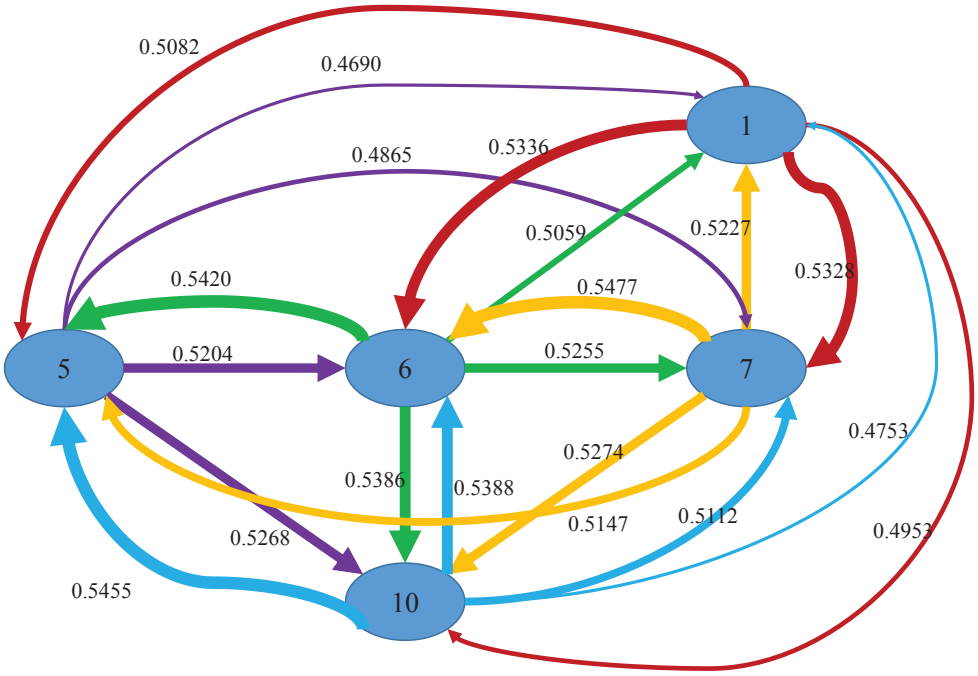


Figure 6.7: Spatiotemporal pattern network for the group of wind turbines.

based on both of these observations, the mutual information based metric is an effective technique to capture the spatial and temporal patterns in wind turbine systems.

Next, we evaluate the effectiveness of the STPN in revealing causal dependencies through wind power prediction. The symbolic and continuous prediction of one wind turbine power is based on the observed symbol sequence emerging from another turbine. According to the procedure of energy prediction described above, Figure 6.9 and Figure 6.10 show the symbol prediction results in which the predicted symbol sequences emerging from wind turbine 5 under the observations of wind turbines 6 and 7, respectively, are compared to the true symbol sequences emerging from wind turbine 5. It is noted that the model is trained using the data from the first half year of 2006, while tested for the second half year of data. From those two plots it can be observed that most of time the proposed xD-Markov machines have a strong prediction capability, while some errors may come from the transient symbols. Moreover, by inspection it can be found that the prediction using wind turbine 6 is slightly better than that by wind turbine 7 as supported by mutual information.

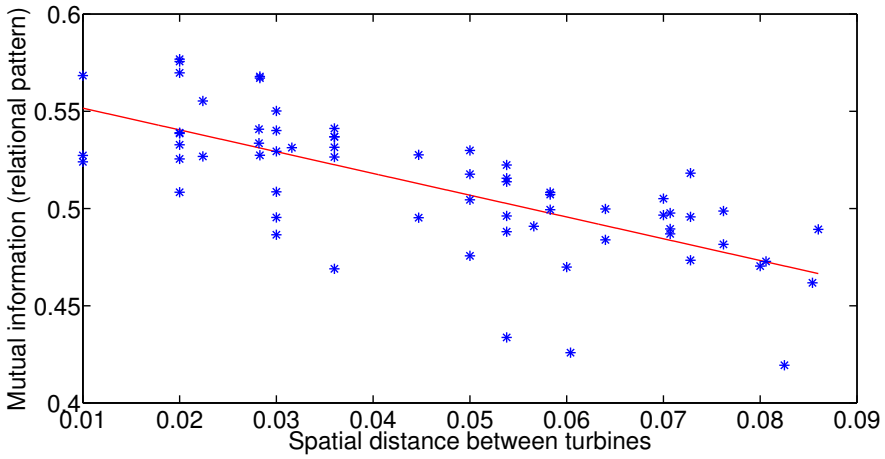


Figure 6.8: A monotonically decreasing relationship for all pairs of wind turbines when spatial distances increase.

Figure 6.11 shows that the mean square error (MSE) is a function of spatial distance between any pair of wind turbines using wind turbines 5, 6, 7, 8, and 9 and it displays a monotonically increasing trend. The prediction capacity in terms of symbols using the proposed STPN has been shown. An example of energy prediction for wind turbine 5 in the continuous domain with the observation of symbol sequence for wind turbine 6 is shown here to validate the energy prediction method.

The plot of Figure 6.12 shows that the major trend in the actual data can be captured quite well and accurately for the continuous domain prediction as the partitioning method MBD is effective in preserving the input-output relation. However, a finer discretization may further improve the prediction result in the continuous domain even though requiring a larger amount of data and increasing computational complexity accordingly.

In the results discussed above, we show that the causal dependencies captured between the neighboring wind turbines using STPN are capable of predicting the wind power production of a turbine using the measured wind speed and wind power for a neighboring turbine. Note, in this framework, we do not use the speed-power curve to predict the power output of a turbine, instead we directly predict the power using only the wind speed and power of a neighboring turbine. However,

we may still need wind speed prediction for a turbine in order to predict the power of its neighboring turbine as this will be quite different from the free stream velocity prediction collected possibly from a different source such as a meteorological forecasting service. While a detail engineering analysis may be needed regarding that in order to transition this research to a commercial deployment, we provide some analysis below as a proof of concept.

Figure 6.13 shows the wind speed prediction of a selected wind turbine (wind turbine 5) using wind turbine 6 using STPN. It can be observed that using STPN can capture the primary trend of the wind speed signal for wind turbine 5 using wind turbine 6 information and the prediction capability can be improved if more symbols are used. Similar results can be seen for other wind turbines as well that are not shown here. Moreover, we show two plots of mean squared error (MSE) which imply the wind speed prediction capability decreases along with distances between different wind turbines. Although the general wind direction is from west to east (i.e., from turbine 5 to turbine 9), we show the prediction effectiveness for both directions i.e., west to east and east to west, as we do not know the specific wind directions and the graphical model learning does not really depend on the directionality of wind flow pattern.

In order to evaluate the proposed scheme in wind power prediction, we compare the prediction performance of the STPN framework with a popular alternative approach, namely, the Hidden Markov Model (HMM) with mixture, which is adapted from HMM to deal with multiple variables. A toolbox compatible with MATLAB (Murphy (2013)) is applied in this context. The results in Figure 6.12 show that the proposed prediction method based on STPN framework outperforms the HMM with mixture under visual inspection. Quantitatively, while the MSE for predicted power using HMM with mixture is 99.885, the MSE for predicted power using the proposed algorithm is 18.953. Therefore, it can be concluded that the STPN scheme, in which causal dependencies between different wind turbines are captured, is an effective technique in wind power prediction.

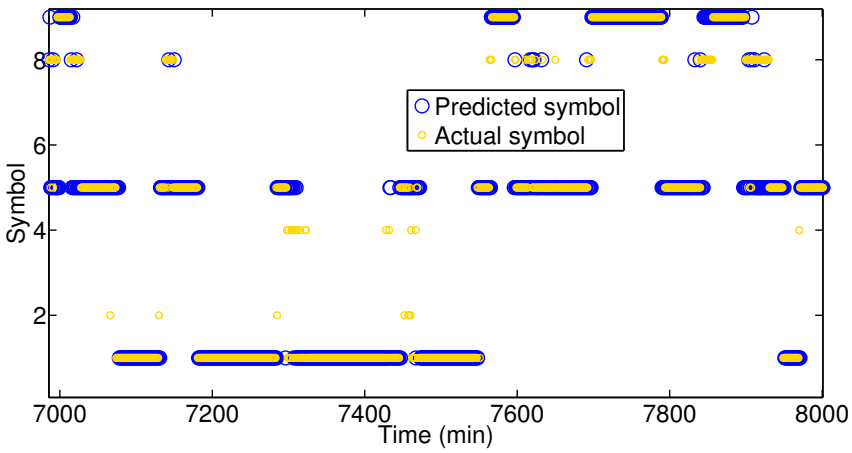


Figure 6.9: Symbolic prediction of wind turbine 5 behavior with the observation of wind turbine 6.

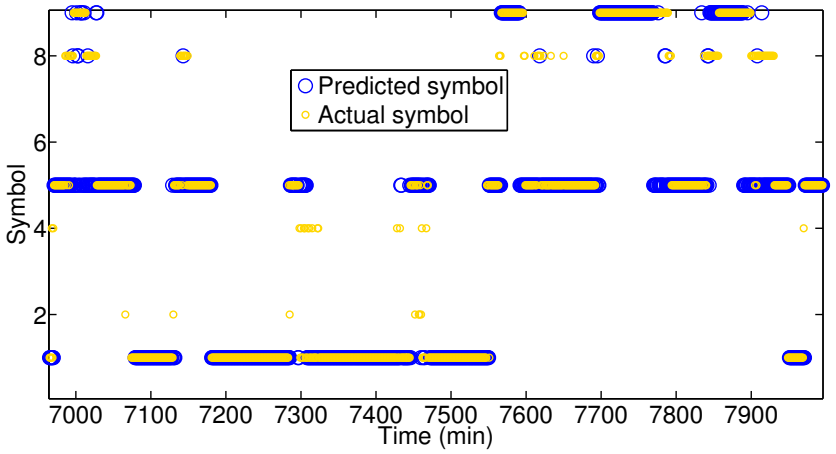


Figure 6.10: Symbolic prediction of wind turbine 5 behavior with the observation of wind turbine 7.

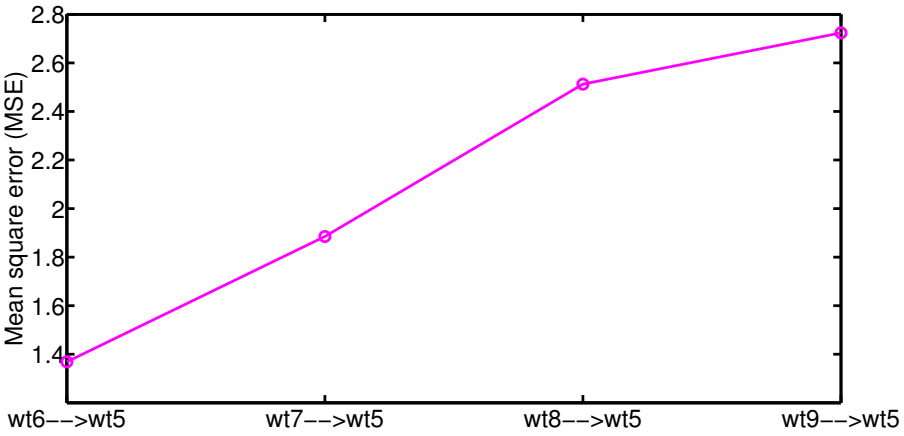


Figure 6.11: MSEs in terms of symbolic prediction of wind turbine 5 power using observation from other turbines: As geographical (spatial) distance increases, MSE increases.

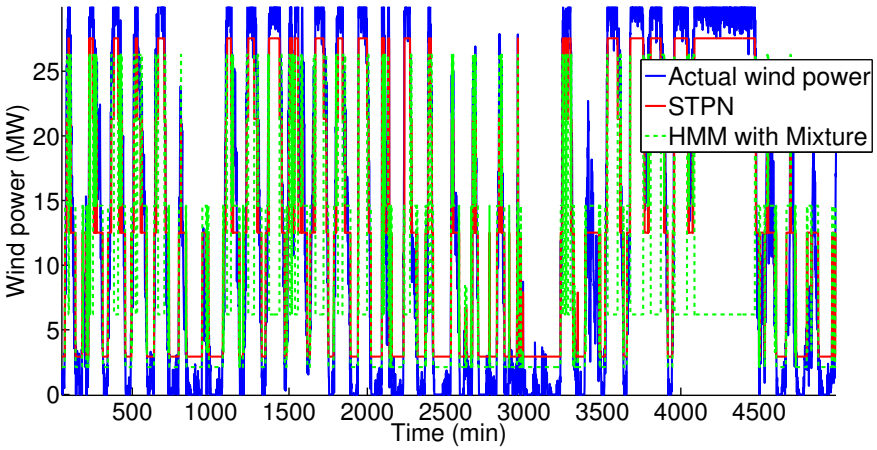


Figure 6.12: Wind power prediction for wind turbine 5 under the observation of symbol sequence of wind turbine 6 using STPN and HMM with mixture.

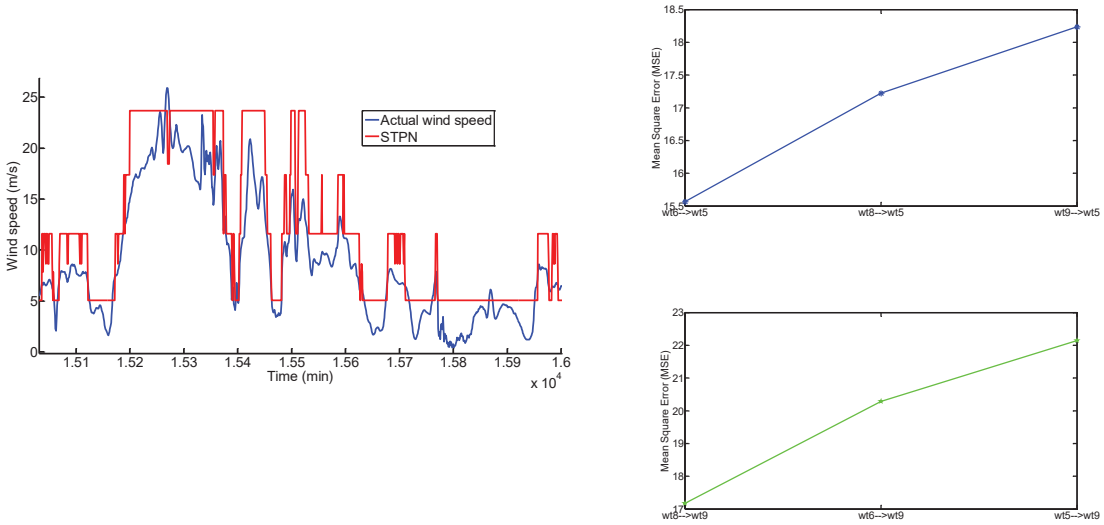


Figure 6.13: Wind speed prediction: Wind turbines 5, 6, 8, and 9 selected for demonstrating the wind speed prediction using STPN. On the left hand side, wind speed prediction of wind turbine 5 is shown using wind turbine 6. On the right hand side, prediction errors are shown for a series of turbines from east to west and from west to east, i.e., wind speed prediction of wind turbine 5 using wind turbines 6, 8, and 9 (from east to west) and wind speed prediction of wind turbine 9 using wind turbines 8, 6, and 5 (from west to east).

6.5 Demand Side: Non-Intrusive Load Monitoring

This subsection presents a second case study based on demand side energy systems; in particular, non-intrusive load monitoring (NILM) of electrical demand with the purpose of identifying electric load components for residential buildings. As described in the above section, the STPN framework is deployed similarly for electric load component disaggregation. In order to best identify the disaggregated energy usage corresponding to each electric energy consuming component from the total energy consumption, convex programming is applied. This step is necessary because for NILM there is no clear input-output relation with the result that–even though the STPN is used in this case study–the results obtained may not be optimal. Here, optimal means that the summation of all load components of residential home energy consumers adds up to the known whole building electricity use. Therefore, for the prediction results by STPN, a convex programming based modification is introduced to achieve the optimal disaggregation.

6.5.1 Problem description

For this case, the data set used for energy disaggregation is based on the Building America 2010 data set available from NREL (Hendron and Engebrecht (2010)). The data is for the hot and dry location of Bakersfield, California with ample examples of heating, ventilation, and air-conditioning (HVAC) energy use in the summer, including lights, appliances (APPL), and miscellaneous electric loads (MELS) along with whole building electric (WBE), which is the sum of all components. The goal here is to apply the measured WBE time series to predict HVAC, LIGHTS, APPL, and MELS, respectively. It is noted that one month of data is adopted where the first three weeks of data are used for training the model, while the fourth week is used for testing and evaluating the model. After the supervised training is complete, WBE is the only known variable.

Convex Programming: Before presenting the prediction results, the convex programming problem setup is formulated for completeness. Suppose that the results obtained by STPN framework are group truth for each part except WBE. Thus, the optimization problem can be expressed

by

$$\begin{aligned} &\text{minimize}_{C_i, i=1,2,3,4} J := \sum_{i=1}^{4} \|C_i - \hat{C}_i\|_2^2 \\ &\text{subject to} \sum_{i=1}^{4} C_i = S; C_i \in \mathcal{R}_{\geq 0}^n \end{aligned} \tag{6.2}$$

where C_i represent the decision variables to be determined, $\hat{C}_i$ signify the prediction results obtained from STPN, S is the known values of WBE, $\|C_i - \hat{C}_i\|_2$ is the Euclidean norm between C_i and $\hat{C}_i$.

The pseudocode of energy prediction based on STPN framework and convex programming is shown as follows. Throughout the rest of analysis, we use STPN+convex programming for reference of the combination of the STPN framework and convex programming technique.

Input: Training data sets $S, C_i'(i = 1, 2, 3, 4)$, depth of D,

Output: Optimal results $C_i(i = 1, 2, 3, 4)$,

Run all of steps in Algorithm 2

Get results by STPN and solve optimization problem in Equation 6.2

Obtain the optimal results $C_i(i = 1, 2, 3, 4)$

Algorithm 3: Energy prediction using STPN + convex programming.

6.5.2 Compared techniques

FHMM 2.4.3 and CO algorithms were ran off-the-shelf as available in the non-intrusive load monitoring toolkit (Batra et al. (2014)) with an exact inference (Ghahramani and Jordan (1997)) for the FHMM. The application of FHMM is accomplished by representing each end use as a hidden state that is modeled by multinomial distribution using $\mathcal{K}$ discrete values, and then summing each appliance meter's individual independent contribution to the expected observation (i.e., the total expected main meter value). The AFAMAP variance of FHMM described in (Kolter and Jaakola (2012)), which includes the trends in the hidden states of FHMM, has also been reported to be effective in the disaggregation task. In the application of FHMM, the number of hidden states is the number of energy end uses, while $\mathcal{K} = 3$ in order to keep the computational requirements low. Combinatorial optimization (CO) (Cook et al. (2011)) algorithm, on the other hand is a heuristic

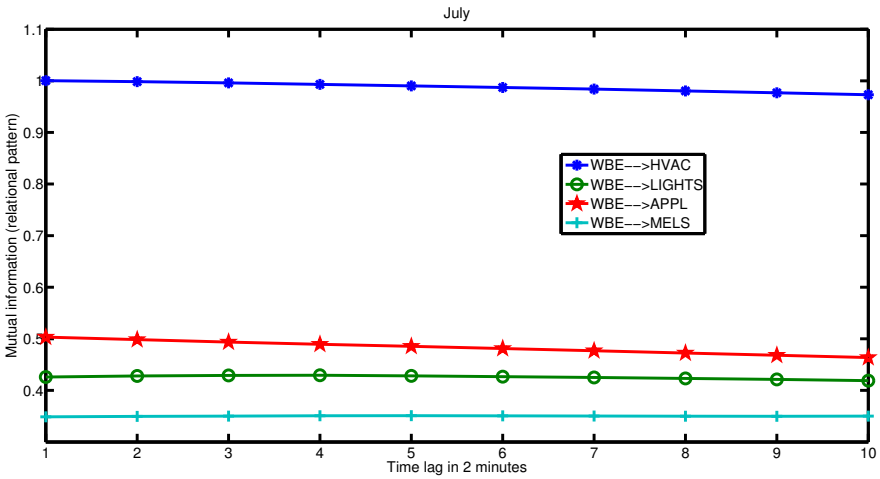


Figure 6.14: Mutual information between WBE and HVAC, WBE and LIGHTS, WBE and APPL, and WBE and MELS with the increment of time lag of 2 minutes in July, 2010.

scheme that attempts to minimize the ℓ_1-norm of the total power at the mains and the sum of the power of the end uses, given either single or multi-state formulation of the sum. The drawbacks of CO for disaggregation tasks are its sensitivity to transients and degradation with increasing number of devices or similarity in device characteristics.

6.5.3 Results and discussion

For validation of the proposed energy prediction approach, the months of April and July are selected to study the prediction performance. As the Building America 2010 data set has 1-hour sampling frequency and three weeks of data are used for training, such scale of data may not meet the data size requirement for the construction of STPN. Building up STPN with insufficient data may result in poor accuracy of causal dependencies between different variables. Therefore, a data reprocessing technique, i.e., upsampling, is applied in this case and the upsampling fold is 30 such that the sampling frequency for the data set is 2 minutes.

First, we study the causal dependencies among these five variables by computing the mutual information. Figure 6.14 shows the variation of mutual information with respect to time lag of 2 minutes to address temporal characteristics. The depth of xDMarkov machine is still 1 such that

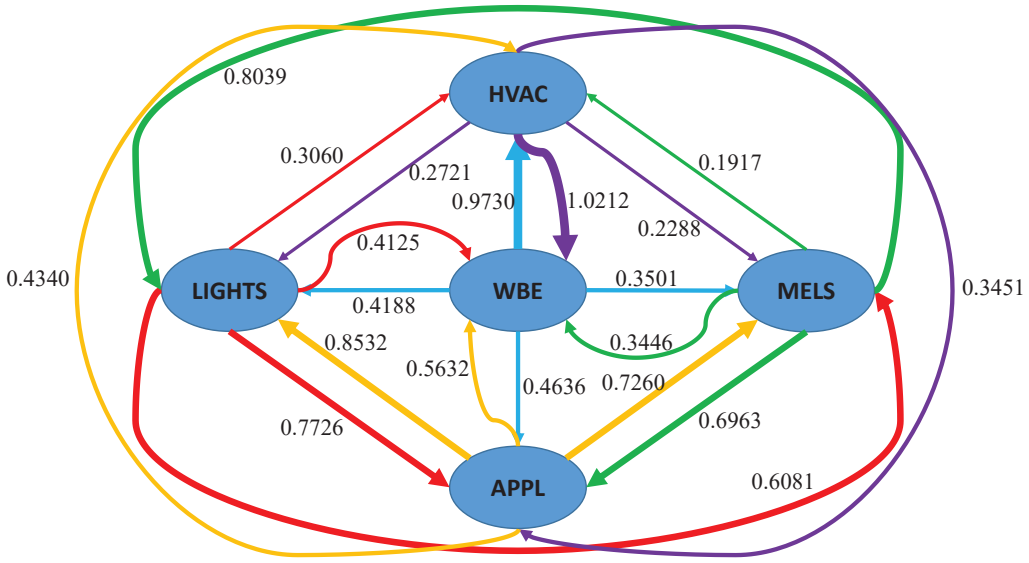


Figure 6.15: STPN using variables, WBE, HVAC, LIGHTS, APPL and MELS in July.

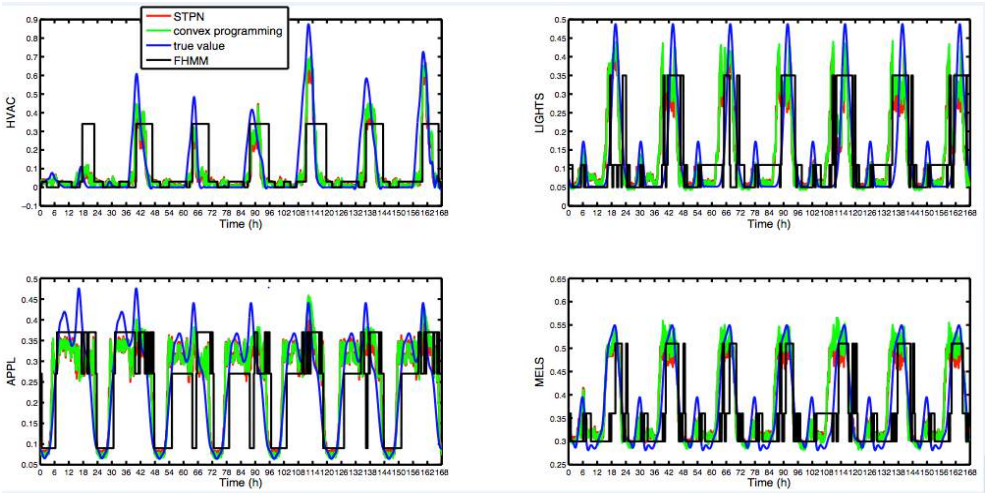
the current symbol of any part of HVAC, LIGHTS, APPL and MELS depends only on the past one symbol of WBE. Different from wind turbine systems, the causal dependencies between WBE and the other four load components have decreased little with an increase of time lag, which reflects that using WBE to predict the energy consumptions of the end uses is temporally robust. However, it also shows that the causal dependency between WBE and HVAC in July is the maximum compared with those between WBE and other load components (i.e., LIGHTS, APPL, and MELS) such that the prediction of HVAC using WBE yields the greatest accuracy.

The results in Figure 6.15 show the causal dependencies quantified by mutual information among all of five variables. It can be observed that the causal dependency between HVAC and APPL is larger than that between HVAC and MELS as well as that between HVAC and LIGHTS. The relationships among LIGHTS, APPL and MELS can be seen to be quite significant due to the causal dependencies obtained in this context. In summary, this relational pattern network captures temporal interactions between different electricity end uses that can be an effective technical tool for energy disaggregation.

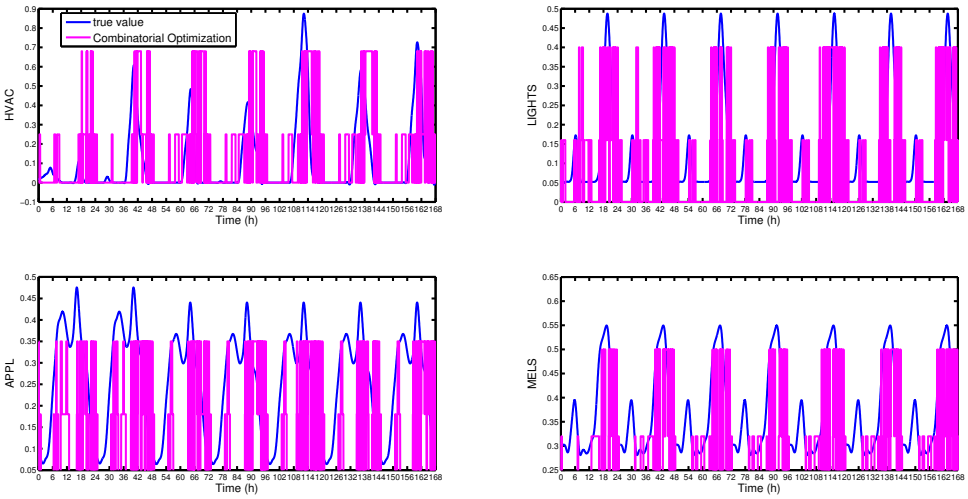
Figure 6.16 shows the energy disaggregation of HVAC, LIGHTS, APPL and MELS using STPN and STPN+convex programming in April. In this month, the energy consumption of HVAC is most significant and accounts for the largest percentage of WBE. A strong prediction capability of STPN can be observed from the plots and based on that, the STPN+convex programming is able to further improve STPN performance, which is attributed to the constraint imposed in the convex programming. It can also be seen from Figure 6.17 that the total energy consumption predicted by STPN without convex programming is worse than the STPN+convex programming results, and the optimal disaggregation appears to be achieved. However, the prediction performance for APPL and LIGHTS is slightly worse than HVAC and MELS because the latter account for a lower percentage of WBE, which is also evident in Figure 6.18.

Therefore, it can be implied that for energy disaggregation, a more accurate prediction can be achieved when one load component (i.e., HVAC, LIGHTS, APPL, and MELS) accounts for a more significant percentage of WBE. It is seen from Figure 6.18 that the prediction for the last two days in the fourth week is worse though it is able to preserve the trend, which may be attributed to the fact that on those two days some transient external factors, such as weather and occupancy, affect the energy consumption. A similar observation that optimal disaggregation is achieved via STPN+convex programming can be made from Figure 6.19. For a direct visual inspection of the prediction capability difference, Figure 6.20 and Figure 6.21 reveal that STPN+convex programming outperforms STPN alone since each part of the energy consumption is predicted optimally. The fact that these two plots show an energy prediction difference by STPN or STPN+convex programming of less than 5% demonstrates the efficacy and effectiveness of the proposed framework.

To show comparison between the proposed method and the current state-of-the-art techniques in literature, we compare the STPN and STPN+convex programming method to FHMM and CO. However, in order to obtain sufficient accuracy of the prediction results, the data set is also upsampled for FHMM with upsampling fold being 1200. Thus the sampling frequency becomes 3 seconds and the number of discrete states used is 3. The energy disaggregation results in Figure 6.16 show that both FHMM and CO perform worse than the proposed method although the predicted



(a)



(b)

Figure 6.16: Energy prediction of HVAC, LIGHTS, APPL, and MELS in April 2010 using STPN, STPN+convex programming, FHMM, and CO separately shown in (b) for better visualization.

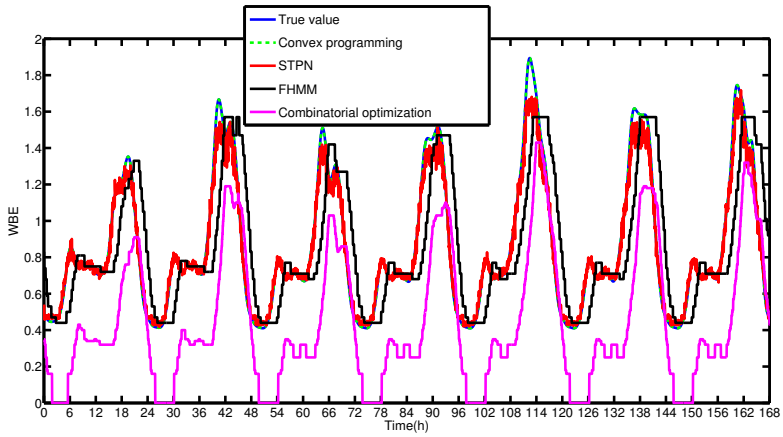
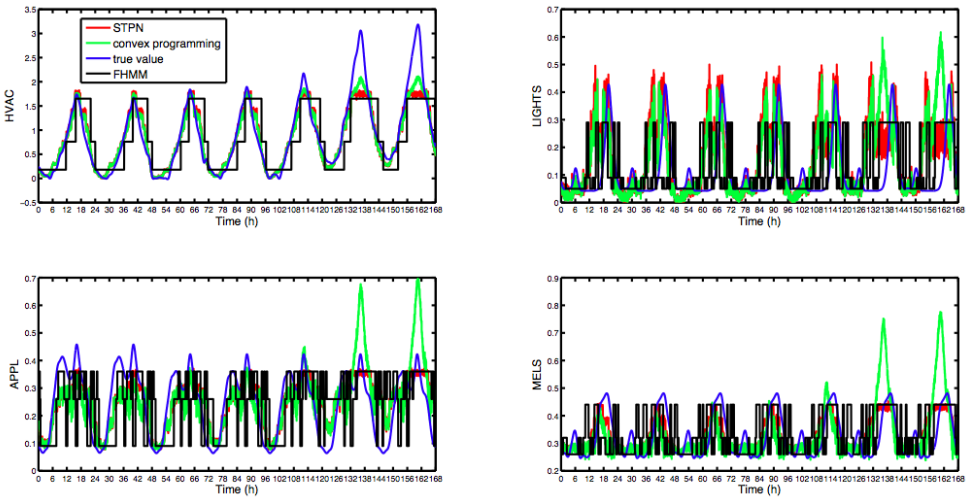


Figure 6.17: Calculated WBE from disaggregated energy values in April 2010 using STPN, STPN+convex programming, FHMM and CO.

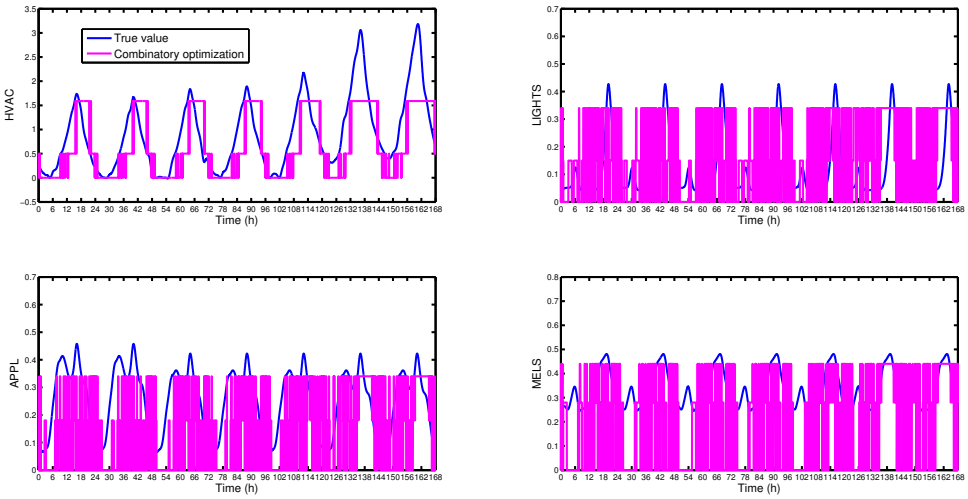
WBE in Figure 6.17 looks quite promising. This effect is explained by the fact that FHMM cannot predict the transient peaks as well as the proposed method and CO is unable to disaggregate the load component adequately.

A similar conclusion is made as for the month of July. From Figure 6.18, it is observed that when the energy curves are more oscillatory, the proposed method is able to outperform FHMM and CO. It can be deduced both from Figures 6.17 and 6.19 that the proposed STPN and STPN+convex programming offer better energy prediction in terms of WBE. Results in Figures 6.20, 6.21 and Table 6.1 quantitatively present the differences among the proposed method (STPN, STPN+convex programming), FHMM, and combinatorial optimization method. It strengthens the conclusion that using STPN and STPN+convex programming yields very promising disaggregation results in NILM. Hence, the comparison among the proposed method and FHMM, combinatorial optimization indicates the effectiveness of the STPN-based energy prediction scheme as a viable tool for energy prediction. We also present the computational effort required for the proposed method, FHMM, and CO.

Remark 6.5.1. Here we also offer the computational time, memory requirements along with accuracy (MSE) in order to compare the performance of different methods. FHMM and combinatorial optimization methods were implemented in ipython notebook for the NILM toolkit (NILMTK)



(a)



(b)

Figure 6.18: Energy prediction of HVAC, LIGHTS, APPL, and MELS in July 2010 using STPN, STPN+convex programming, FHMM, and CO separately shown in (b) for better visualization.

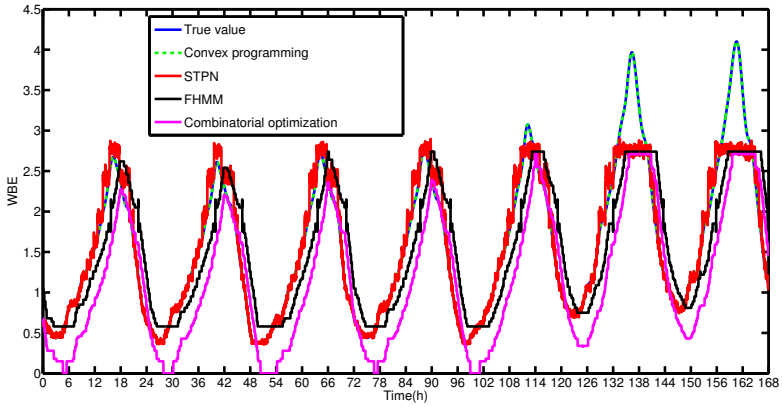


Figure 6.19: Calculated WBE from disaggregated energy values in July 2010 using STPN, STPN+convex programming, FHMM and CO.

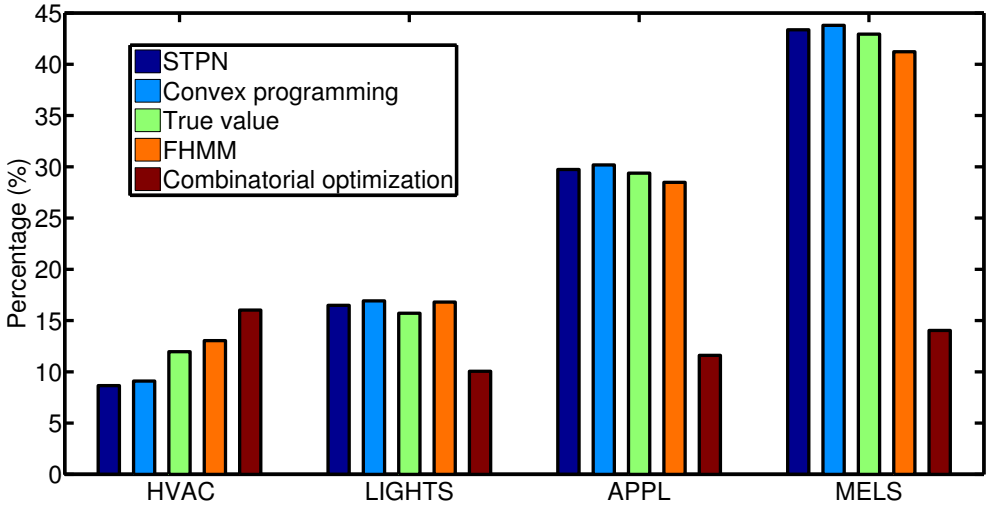


Figure 6.20: Energy prediction difference of HVAC, LIGHTS, APPL, and MELS in April 2010 among STPN, STPN+convex programming, FHMM and CO.

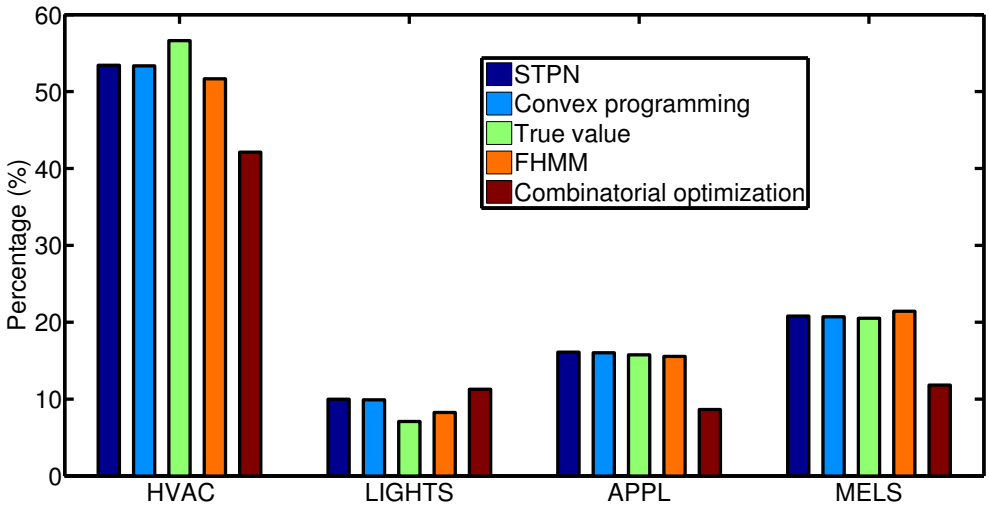


Figure 6.21: Energy prediction difference of HVAC, LIGHTS, APPL, and MELS in July 2010 among STPN, STPN+convex programming, FHMM and CO.

Table 6.1: Computational information for different methods in April.

Method	Time (s)	Memory (MB)	Accuracy (MSE)
STPN	28.74	962.00	0.0072
STPN+convex programming	369.64	2756.00	0.0070
FHMM	38.10	798.67	0.0163
CO	11.25	769.37	0.0564

while STPN and STPN+convex programming were implemented in the MATLAB environment and CVX package (Grant et al. (2008)). The results in Table 6.1 show that STPN can spend less time than FHMM while more memory is required as the number of states for STPN is greater than FHMM in this case. STPN+convex programming approach demands more computational time and memory to run the whole process due to the optimizing iterations. FHMM and CO use less memory compared to the proposed schemes. However, in terms of accuracy, the STPN outperforms FHMM and CO approaches as shown in Table 6.1. The MSE of FHMM is more than two times as that of STPN. Moreover, STPN+convex programming is able to further improve the accuracy obtained from the STPN framework. In summary, energy prediction based on the STPN framework may be

an effective method of energy prediction. Note, the FHMM and the CO codes used here are part of a well–optimized toolbox and we expect that similar code and platform optimization can bring our proposed methods to a comparable level in terms of memory and time complexity.

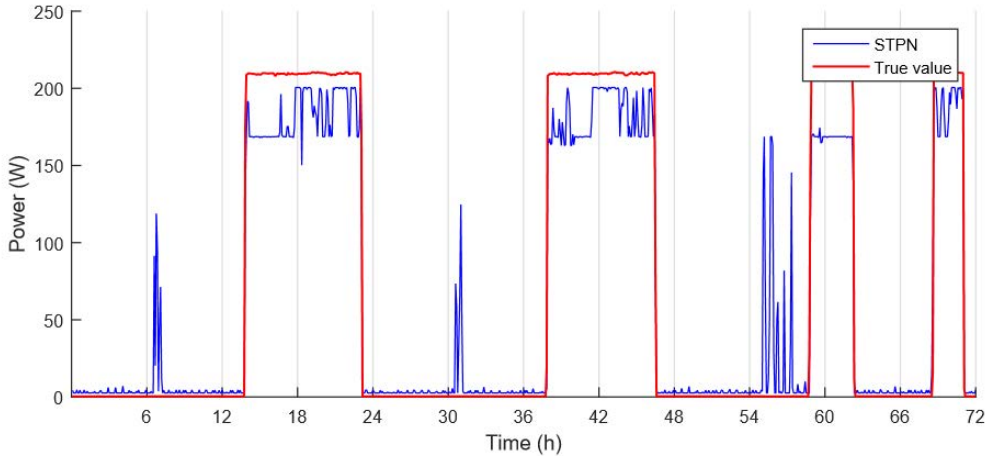
Case study on disaggregation using high-frequency data set: The previous test data is sampled every hour, and an upsampling technique is applied to yield more data points to apply the proposed method. To further test the STPN framework on high-frequency data set, two data sets are considered, the Almanac of Minutely Power Dataset (AMPDs) (Makonin et al. (2013, 2016)) and Electricity Consumption and Occupancy (ECO) data set (Beckel et al. (2014); Kleiminger et al. (2015)), which are sampled every minute and second respectively. The latter data set is used in this work to validate the proposed algorithm for high-frequency data. For more details of the data set, please refer to (Beckel et al. (2014); Kleiminger et al. (2015)). Using the collected data during January 2013 at household 2, the validation is carried out with the same settings as for the previous cases, where the first three weeks of data are used for training the model, while the fourth week data are used for testing. The disaggregation results of appliance 12 (stereo) in the testing week are shown in Figure 6.23, and the performance of the three methods (STPN, FHMM, and CO) on each appliance is listed in Table 6.2. Results show that the STPN framework outperforms FHMM and CO.

Table 6.2: Performance (MSE) of STPN with comparison to FHMM and CO.

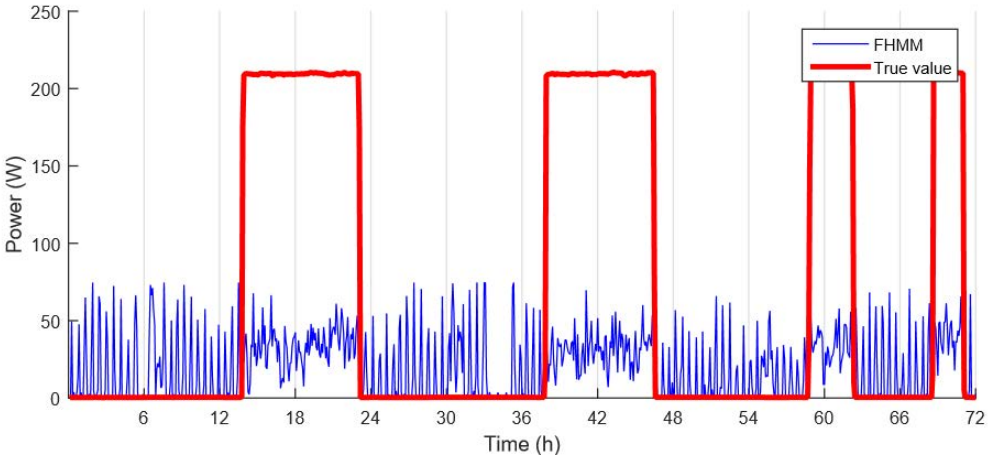
ID	FHMM	CO	STPN	ID	FHMM	CO	STPN
1	2.6	4.9	1.3	7	2.89e4	1.90e4	0.42e4
2	4.12e4	3.89e4	1.94e4	8	3558.0	3641.9	1245.6
3	946.1	1771.6	18.6	9	369.9	915.5	225.4
4	2065.7	5208.9	1724.0	11	4720.0	4166.2	925.2
5	7580.9	8871.3	1303.6	12	735.7	866.5	154.4
6	1401.0	5102.8	1291.2				

*For appliance 10, performance is not calculated as all values equal to zero.

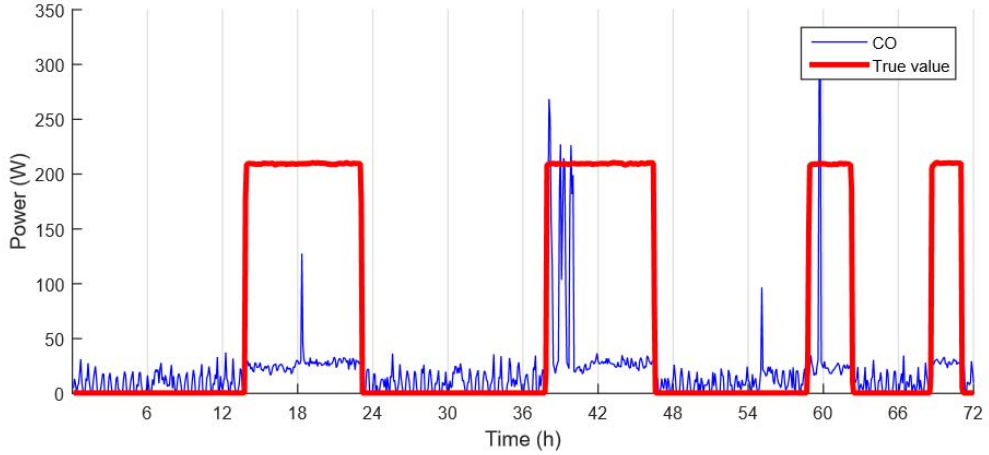
Remark 6.5.2. The case study with ECO data set is explored to show the effectiveness of the proposed STPN framework in high-frequency data. The memory and time are not compared here,



(a) STPN on entertainment–appliance 5

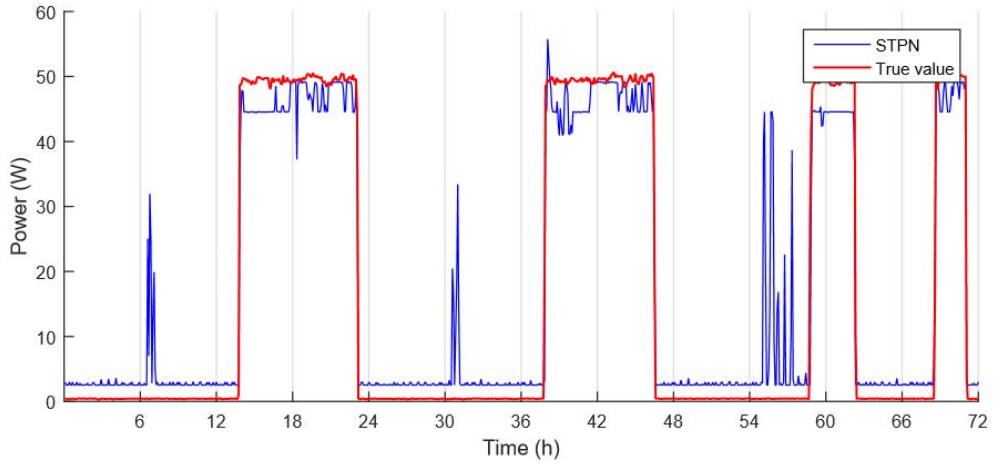


(b) FHMM on entertainment

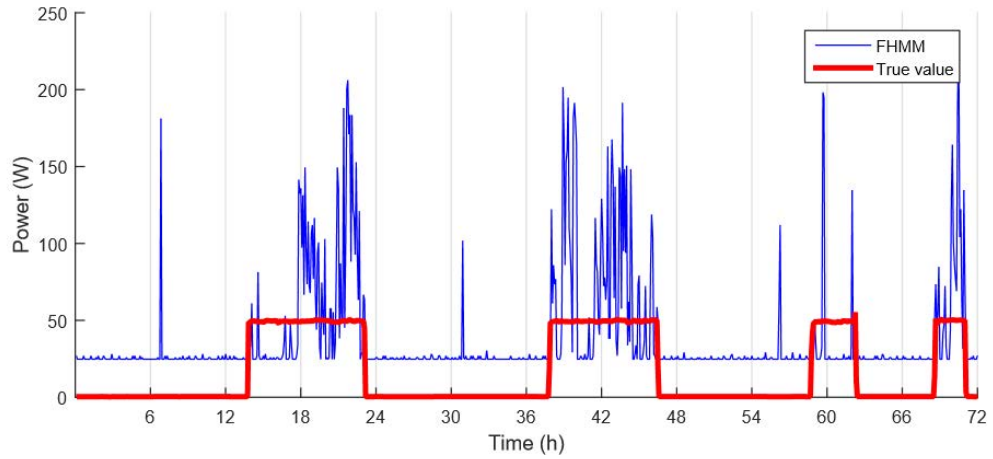


(c) CO on entertainment

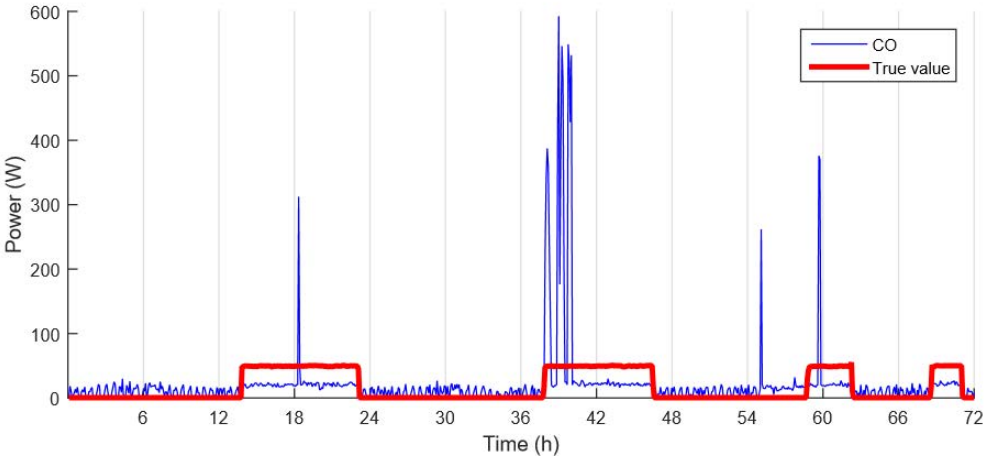
Figure 6.22: Disaggregation results using STPN, FHMM, and CO. The results are shown at every 5 minutes for better resolution (average value from 300 predictions in 5 minutes).



(a) STPN on stereo–appliance 12



(b) FHMM on stereo



(c) CO on stereo

Figure 6.23: Disaggregation results using STPN, FHMM, and CO. The results are shown at every 5 minutes for better resolution (average value from 300 predictions in 5 minutes).

one important reason is that the NILMTK takes several days to generate the results for FHMM and CO.

6.6 Summary

In summary, a novel data-driven spatiotemporal pattern network (STPN) has been employed to predict energy production/consumption. The spatiotemporal extension built upon SDF, using xD-Markov machine learned to capture causal dependencies between dynamic sub-systems. The network is validated by wind turbine power prediction as well as the residential electric energy disaggregation (demand side energy) using the Building America 2010 data set from National renewable Energy Laboratory (NREL). STPN captures salient spatiotemporal features and achieves high-accuracy prediction. STPN plus convex programming outperform state-of-the-art techniques in Non-intrusive Load Monitoring (NILM).

CHAPTER 7. FEATURE EXTRACTION FROM SPATIOTEMPORAL DATA – 3D-CONVOLUTIONAL SELECTIVE AUTOENCODER (3D-CSAE)

In Chapter 4, a convolutional selective autoencoder (CSAE) was introduced to analyze 2 dimensional images, hence here termed a 2D-CSAE. In this chapter the 2D-CSAE is extended to a 3-dimensional (3D) architecture to characterize the spatial and temporal scales present in flame videos.

7.1 Background and Motivation

In jet engines and power generation systems, strict NOx emission laws have resulted in shift of large chunk of engine design issues to the domain of engineering control (Samad and Annaswamy (2011)) to contain the effects of combustion instabilities. Therefore, the ensuing approach to mitigate engine-related pollution hazards has been in favor of leaner (i.e., increased air-fuel ratio), premixed equivalent mixture that are atomized in the combustion chambers (Huang and Yang (2009)). Consequently, a system that is burning the mixture normally (i.e., in a stable state) experiences some shorter intermittent time scale, where it transitions to unstable state resulting in blowout at those time scales. The main problems associated with such designs are the instabilities which usually become more prominently sustained in longer time-scales. Spreading flames have also been found to exhibit similar instabilities when the environmental factors (temperature, wind or topography) couple with the fire (Fox and Whiteside (1987)). In addition, the acoustics, vibrations, chemical kinetics etc. in turbulence couple with the flames in the combustor, together leading to combustion instabilities. A significant improvement for the application is to return to the image space by detecting the regions in the flame that visually characterize the properties of coherent structures. Having implemented a detection-type network on the problem in Section 4, the results of such network (i.e., the structures) have added appealing visual explanations for engineers about

what instability regions in assumed stable frame look like. This in a way extends beyond classifying the flames via a scalar metric in previous approaches (Simonyan and Zisserman (2015)). However, the results of the approach were not justifiable for understanding the dynamics of the thermo-acoustic instabilities.

Our approach is to attempt to characterize the dynamics of the structure of instability from known unstable frames, as well as a corpus of expert identifiable stable frames which are extracted off the videos that were collected from a laboratory scale swirl-stabilized combustor. The combustor experiments and setup that produced the videos have been described in Chapter 4. In addition to consideration of early identification of the intermittent structures present in flames that are very similar, with subtle differences, to stable frames, we model how the intensity (or energy) is spread around such intermittent frames.

The main interests here are to (*i*) derive several short videos (or streams (Simonyan and Zisserman (2015))) formed by grouping together short bursts (sequential windowed burst of frames) with temporal over-lapping strides from the limited availability of hi-speed flame video, (*ii*) extract low dimensional features that jointly characterize individual frames and then streams of both stable and unstable flame respectively, (*iii*) learn the features of understudied combustion instability in a robust way to aid transfer learning to extracting unsuspecting regions of coherent structures in unseen frames, (*iv*) characterize these features in a hierarchical manner up until the most basic low-dimensional manifold of the structures, (*v*) detect the early signals of the onset of instability features, (*vi*) understand how the structures are evolving in the short sequential volumetric frames, (*vii*) overcome the known effects of combustion noise that arises from volumetric expansion due to reactions of the mixtures and perhaps the gradient in products' flow, (*viii*) exploring the applicability of 'neuro-scientists' hypothesis of 'where' and 'what' pathways (Goodale and Milner (1992)) in spatiotemporal dataset training. To achieve the state aims, we designed and developed a novel 3D convolutional autoencoder (3D-CSAE) (improving the convolutional neural network) that is trained to act selectively, i.e., to filter examples from one flame regime and learn the structures from the other flame regime, to (*i*) conditionally capture the correlation of the images at multiple

spatial embedding (or scales) of the each image through a hierarchical systems and (*ii*) jointly localize, at such scales, the evolution of the coherent structures from each frame in a short sequence to the next.

For temporal sequence modeling within a deep learning framework, long short term memory (LSTM) (Wu et al. (2015)) of a recurrent neural network (RNN) (Graves and Schmidhuber (2005)), multiple CNNs (Simonyan and Zisserman (2014)) are plausible ways to model the long term dynamics present in a spatiotemporal data, such as the sequential frames of the available dataset. These architectures are not able to provide explanations beyond clues on actions taken. 3D networks on the other hand have not received so much attention in the community. Despite 3D-CNNs' identified potentials for localizing the relationships in the 3rd dimension, the networks have not enjoyed so much attention, probably due to the associated computational requirement. To the best of our knowledge, the 3D convolutional autoencoder architecture is the first of its kind for the current identification problem. However, a 3D Fully Convolutional Network (FCN), having different configuration has been utilized for vehicle detection from 3D Lidar (Li et al. (2016)), while a similar network was applied for a point cloud detection of vehicles (Li (2017)). We have taken an alternative approach for proper scaling of the network parameters, and this will be evident during experimentation. Other previous concept that share similar formulation to our approach have been explored mainly for video classification (Karpathy et al. (2014)) and human action recognition (S.Ji (2010); Simonyan and Zisserman (2014)). Simonyan and Zisserman (2015) specifically referred to the temporal short time window as optical flow, and designed a multiple stream CNN, spatial and temporal ones, for improved performance. Therefore, rather than implementing multiple streams of the CNN separately, we model a third dimension to localize the temporal frame sequence. The spatiotemporal decomposition of the frames used for the human recognition was related to two-streams development. Here, such decomposition is extended beyond performing a classification task to our end-to-end identification task, although we also evaluate the consistency of the middle (also bottleneck or coding) layer of our network for classification task. In the current model also (as in Akintayo et al. (2016b)) selectivity has been included. Therefore, in the light of our objectives,

this work trains stacks of short videos – volumetric frames to characterize the early detection of instability in unseen videos called protocols (Sarkar et al. (2015c)) and also discover more intuitive, consistent coherent structures evolution in the stable region. The experimental setup that facilitated collecting the videos as well as the procedure for that purpose has been described in Subsection 4.4.3.1.

7.1.1 3D temporal stacking of 2D frame sequence

Suppose a typical flame video $\mathbf{V}$ taken from the combustor is 'deframed' to the set $\mathbf{V} = \{I_1, \cdots, I_t, \cdots, I_T\}$, where each $I_t \in \mathcal{R}^{H \times W}$ is a frame image in the dimension space where, H-height and W-width, and T is the total number of frames in the sequence. A spatial 2D-CNN simply treats individual frame as singly occurring by randomly selecting a group in the batch-wise run of the algorithm. Our 3D-CSAE is designed to be capable of correlating spatial scales as well as multiple short time scales. The first step is therefore a temporal stacking of the frames. Figure 7.1 shows example frames that have been stacked together to form the shown volumetric frames. Note that the datasets here have been color-mapped to indicate the intensity of the structures on the frames. To achieve that using the sequential frames of the video, we parameterize the local dependency of temporal frames by N in order to experiment on multiple values that yield the most feasible number of sequential frames. Also, we augment the dataset by retaining, to sufficiency level, the spatial correlation (like the 2D case) through overlaps in the time scales. The resulting volumetric examples from stacked and overlapped 2D frames has the expression, $V^{(1i)} = \{I_1, \cdots, I_N\}, V^{(2i)} = \{I_{1+k}, \cdots, I_{N+k}\}, \cdots, V^{(ji)} = \{I_{1+(j-1)k}, \cdots, I_{N+(j-1)k}\}, \cdots$.

Where, j is index for stacked volumetric frames, N the number of stacked frames in the volume and k is the temporal gap between stacked volumetric frames. Each example $V^{(ji)} \in \{\mathcal{S}, \mathcal{U}\}$, where $\mathcal{S}$ is the set of all stable frames and $\mathcal{U}$ is the set of all unstable frames for distinctness. Few supporting formulations have been termed optical flow stacking where the our temporal window is termed an optical stream (bidirectional optical stream by Simonyan and Zisserman (2015)) which considers][1:2N] sequence (in forward and backward) unlike our [1:N] frame sequence. Dense

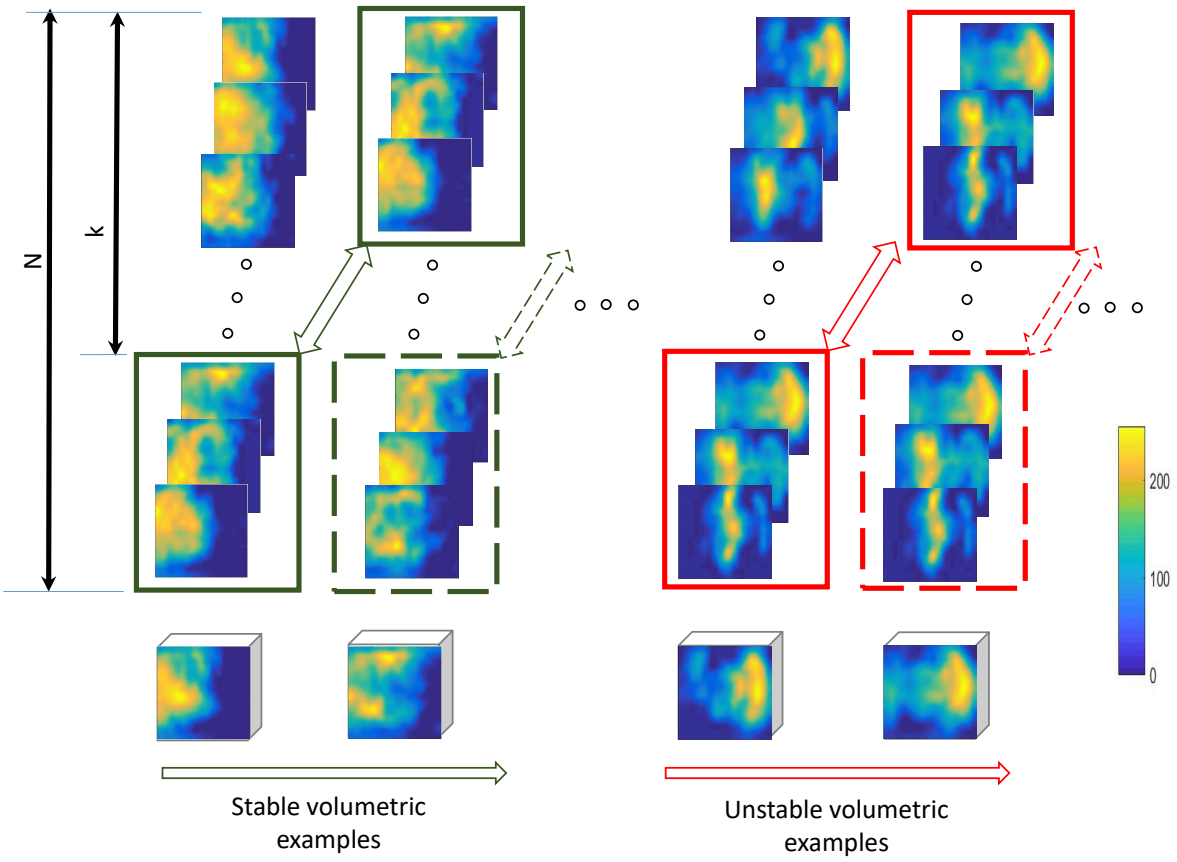


Figure 7.1: Volumetric dataset example of short time set of stacked frames N and overlap k with neighboring frames for stable and unstable region respectively. Note: False colors are used to aid visualization.

trajectory stacking using recurrence relations have also been employed (Wang et al. (2011)). The approach in this work differs due to our need to determine N. Also, rather than having to stack multiple parallel networks, we train a single network for the detection purpose, while checking the consistency of its coding layer classifier. The appealing structure of our stacking scheme is that it gives 3D-CSAE the ability to model both a 2D-CSAE (Akintayo et al. (2016b)) and temporal streams of such sequences in a single network. It is noted here that false colors have been used in Figure 7.1 and subsequent figures in this chapter for a clearer distinction of the structures on the frames.

7.2 Approach

All the volumetric frames are normalized before the start of the experiment by simply removing the overall mean and dividing the resulting values by the standard deviation. Mathematically, this is equivalent to $V^j = \frac{V_j - \mu(\mathbf{V})}{\sigma(\mathbf{V})}$. The data analysis approach that we deploy for the detection and evolution is a 3D convolutional selective autoencoder. The technique explores multiple number of convolutions at multiple scales and dimensions of the volumetric frames to learn the most representative features for the stable and the unstable frames, devoid of feature-crafting.

7.2.1 3D convolutional autoencoder (3D-CSAE)

Here, we describe the 3D convolutional selective autoencoder layers that make up the network and background for their choices. The network provides plausible ways to explain the underlying physics behind evolution of combustion instabilities in combustors from the available hi-speed video observation. Our network makes use of the prepared volumetric images that have been preprocessed as described in data collection and preprocessing (Subsection 4.4.3.1). The layers of a 2D (or 3D) CNN (Krizhevsky et al. (2012b); LeCun et al. (1998a)) are the well known convolution, pooling, dropout and dense layers at the fully connected stage. However, for our visualization of the output in input-similar space, the end-to-end convolutional layers is decoded via the unpooling layers (Akintayo et al. (2016b) or simply more convolution layers (Li (2017)). While such networks are

standard, we describe the approach to show consistency with the tasks of detection. The convolution layers in Figure 7.2 are initialized with a number of uniformly random 3D kernels. These scan through batches of volumetric frames' inputs by performing the weighted sum of scaled and shifted product of kernels and input to produce the feature maps (also layer output). Large bank of kernels (output channels) that bears similarities with dictionaries (Elad and Aharon (2006)) can be learned in the process. The pooling layers are simply reducing the computational burden (introduced by several kernel banks) by selecting only maximum activation (maxpooling) from the output of the convolution layers. They also improve the posterior probabilities which would otherwise be spread over several units, thus reducing the number of training epochs. The fully connected layers are similar to the autoencoder (Vincent et al. (2008)), in which input (i.e., vectorized units) are compressed to lower dimensions the hitherto high dimensional outputs of the convolution layers. In the fully connected layers, parameters such as the weights and biases are also learned.

The implications for our application are that: at each convolution layer, a correlation-like evaluation is done. Assume that we desire to come up with Y number of activated features, $V^{(Y)}$ by randomly sampling for $y \in \{1, \cdots, Y\}$ kernels, $K^{(y)}$, then convolving and summing with X number of input features, $V^{(x)}$ in the sequence $x \in \{1, \cdots, X\}$.

$$V^{y} = \sum_{X} \left[\sum K^{(y)}(k_j, k_h, k_w) \times V^{(x)}((j-1)k_j : jk_j, (h-1)k_h : hk_h, (w-1)k_w : wk_w) \right]_{\substack{w=1:W-k_w+1, \\ h=1:H-k_h+1, \\ j=1:J-k_j+1}} \tag{7.1}$$

With our current 3D-CSAE, the correlations are done at multiple spatial and temporal scales, while the layer-wise network ensures that such correlations are also performed at several transformations (dimensional equivalents) of the input volumetric flames. The space of the outputs can then be transformed by a rectified linear unit (ReLU) activation (Glorot et al. (2011)), which attenuates all the negative units that are most likely not useful. The fully connected layers are efficient for showing how the the layers' posteriors discriminates the ground truth present in the known training examples. Finally unpooling reverts the image to its original dimensions, with more filter learned in subsequent convolution layers. Dropout is the last important layer for most architectures which

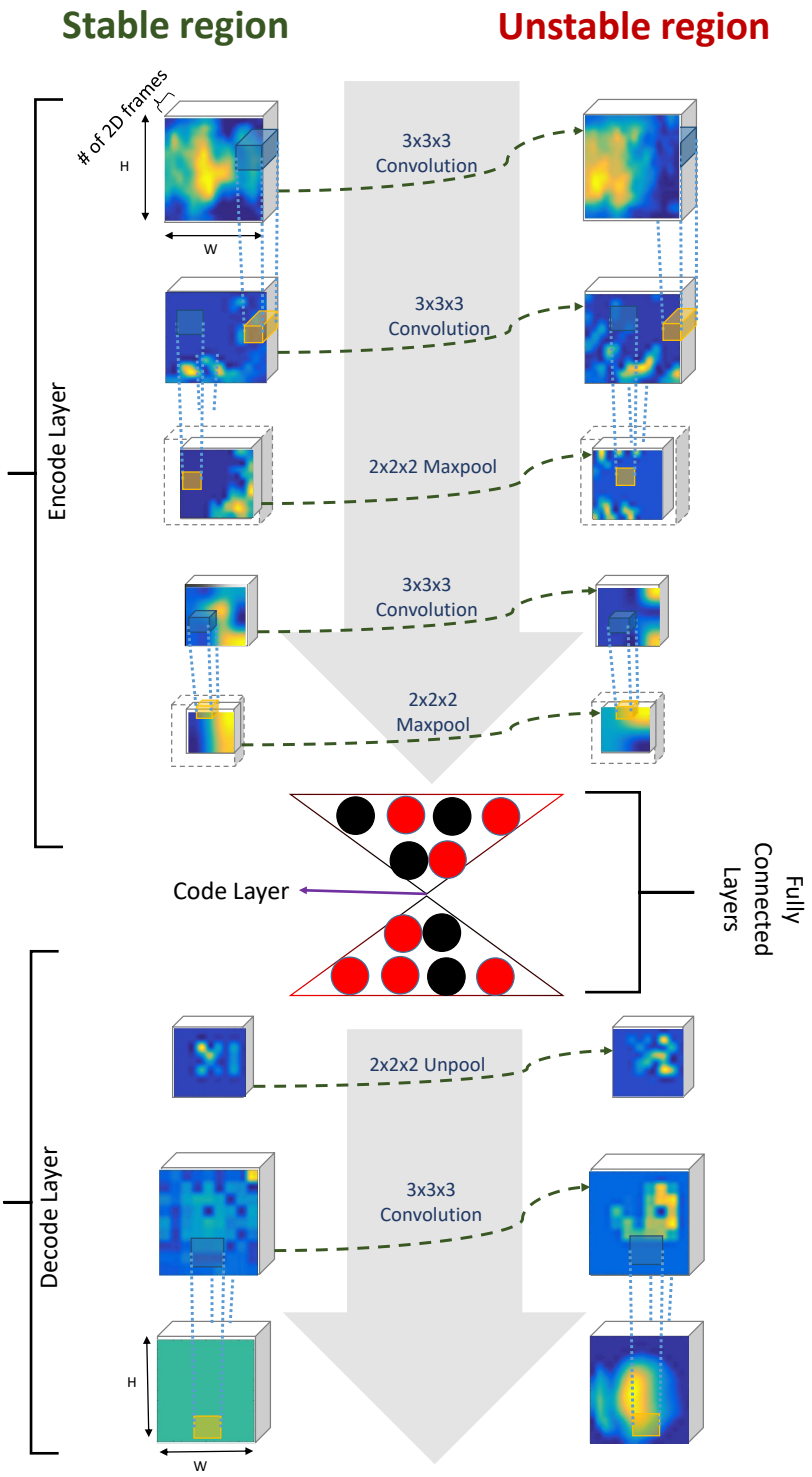


Figure 7.2: 3D Convolutional selective autoencoder framework that accepts bursts of volumetric frames for exploring the short ranged relationship between 2D frames in a windowed form and spatial relationship between adjacent volumes, and activating volumetric features' examples in training for each region. Dropout=0.3 in corresponding dropout layers (Asterisks (*)).

we included to simulate some randomness into the system at each run, thus trying to simulate some time-averaging that has been found useful for analyzing chaotic turbulence. The error in masking, or not, of the appropriate regions are backpropagated to modify the weights and activations of the feature maps at each run of the algorithm. The summary of Figure 7.2 is: pass the input volumetric frames in batches through the network, obtain layer-wise outputs from the network as features and learn to back-propagate the error between the forward output and target number image labels. To reduce the error, the parameters are slightly adjusted in the optimization problem.

7.2.2 Network parameters and structure

We obtain best-performing networks when all the hyper-parameters are jointly selected rather than individually, thus the space requires choosing from a wide range of hyper-parameters. For our architecture size, the dimensions of the images are originally (H $\times$ W) = (237 $\times$ 250), but are resized to (64 $\times$ 64) in order to reduce the computational requirement. The resulting images are then temporally stacked with appropriate striding dimension. The chosen dimensions for stacking and striding are related to the ability to fit through our available compute nodes because there is the curse of dimensionality associated with 3D networks. As shown in Figure 7.2, the network has 13 layers, 4 convolution layers intermixed with 2 maxpooling layers, 2 dropout layers (Hinton et al. (2012)) and 1 unpooling layer. The fully-connected layers consists of 3 multi-layer perceptron with a decoder layer, a code layer and an encoder layer. Thirty percent of the maxpooled activations are dropped out between the 3rd and 5th layers and another between the 6th and 8th layers. Each of the convolution, maxpooling and unpooling layers uses between 32 and 128 ($3 \times 3 \times 3$) filters, just as Simonyan and Zisserman (2014) identified 3×3 as best choice for the first 2 dimensions of CNN kernels. The nonlinearity activation for all the convolution layers is the already introduced ReLU.

7.2.3 Training

Here the parameters of the model such as the kernels of the convolution layers, the weights and biases of the fully-connected layers of the model are learned as the data moves across the network from the input to the output. We take a simple example of an input volumetric frame to illustrate how the algorithm trains on the dataset. Alongside the network selectivity training criteria of masking the stable volumes and allowing the unstable volumes, we get a feedback from the encode layer of the network. The feed back is in terms of the distribution of the encode layer, its consistency with the training examples and the ground truth provided by the experts. These factors will form the bases for our discussions of the experiments. Let $V^{(jo)}$ be the last layers activation for the input, $V^{(ji)}$ the error function is formulated from the 3D-CSAE selectivity training criterion as follows,

$$f_{3D-CSAE}(V^{(ji)} \in \mathcal{U}) = V^{(jo)} \tag{7.2}$$

$$f_{3D-CSAE}(V^{(ji)} \in \mathcal{S}) = 0 \tag{7.3}$$

Since this study is not intended to experiment on the optimization techniques of deep learning since that topic forms another research area (Dauphin et al. (2014); Jiang et al. (2017a)). We simply hold the learning rate constant at $\frac{1}{1000}$, used in our best performing optimization routine, RMSProp (Dauphin et al. (2015)) that is an online method for non-stationary datasets, and for the detection problem, we use a mean square error optimization function between the final activation and the input data for individual units,

$$min\ f := \sum_{H \times W} (V^{(jo)} - V^{(ji)})^2|_{V^{(ji)} \in \mathcal{U}} + \sum_{H \times W} (0 - V^{(ji)})^2|_{V^{(ji)} \in \mathcal{S}} \tag{7.4}$$

f is the objective function that is optimized, while the optimization error is back-propagated to change the weights and biases in the direction of error reduction. Additionally, we desire a training that filters out stability quality in the unstable region, such that the structures are homogeneous, rather than simply learning the identity present in the ground-truth data. Therefore,

the discriminability of the unfiltered results is desired to be reduced than what is present in the ground truth labels Figure 7.2. Training was done in the Keras (Chollet (2015)) platform running on Theano (Bastien et al. (2012)) backend. Due to the scale of the experiments required to obtain the optimum hyper-parameters, the network was trained using a cluster of 4 relatively high-end Tesla $P40$ graphics processing units (GPUs), each having 3840 NVIDIA® compute universal device architecture (CUDA) cores and with 1.531 GHz clock speed. The hardware provides huge speed-up in the algorithm training, but testing of a selected model can be done on a low-end CPU. Typical examples, for each region, of the feature maps learned by the network after training are highlighted by the sides of the network in Figure 7.2.

7.2.4 Experiments

Here, the details of experiments conducted to determine the hyper-parameters that yields the best results are provided. Seventy-five percent of all the training dataset that results from the streams and augmentation are utilized to train the algorithm while 25% are used to validate the results as well as set the instability threshold. Experiments sequence was initially started with an over-complete (or excess) banks of kernels (i.e.,128), i.e., where there are many possibly sparse kernels (due to the increased dimensionality in 3D compared to 2D), but are still efficient enough with our GPU capability. The order of optimizing the parameters are a major aspect of our experiment.

Our experiments begin with the same configuration as that by Akintayo et al. (2016b) (i.e temporal window and no overlap ($N = 1, k = 1$)), where our 3D-CSAE algorithm is validated to be consistent with the results (Akintayo et al. (2016b)). Steps of N or k in multiples of twos are utilized, with the constraint that $N \geq k$, still having exponential ways to combine them in the implementation. The simultaneous 'best' N and k found are held constant in subsequent experiments. Note also that Akintayo et al. (2016b) had selected 10 classes as the best result, despite the notion that there are 2 classes in the training dataset, we therefore run ours from 2 to

10 in steps of 1. Then, the kernel (filter) bank was reduced from 64 to 16 in order to derive the non-sparse features from the dataset.

7.2.5 Evaluation metric and threshold

The evaluation metric for all the frames is the image KL-divergence (Akintayo et al. (2016b)). Based on the envisaged effectiveness of the architecture in clearly distinguishing the unstable flames and intermittent coherent structures on the stable image frames, the threshold here is taken as the average of the 95% CI of filtered validation examples in the stable region and the 5% CI of the corresponding filtered examples in the unstable region – average threshold. Mathematically, the expression is: $\frac{1}{2}(95\%CI(3DCSAE(\text{stable validation frames}))+5\%CI(3DCSAE(\text{unstable validation frames})))$.

7.3 Results and Discussion

It is discovered that at the same hyper-parameters as the 2D-CSAE, the 'best' values are $N = 32$ or 16 and $k = 4$ or 2 respectively. The latter pair being more favored in the coding layer entropy for both regions, but more computationally burdensome and less homogeneous in the unstable region, while the former benefits from more dataset (i.e., (N=16, k=2) had 31486 while (N =32, k=4) had 31486 volumetric examples). The performance comparison in Figure 7.3 shows the justification for a 3D-CSAE where 3D-CSAE in Figure 7.3(b) shows less noise in the stable region and better distinguishes regions than Figure 7.3(c) in 2D-CSAE.

7.3.1 Visualization of layer-wise class activations

We use the best set of hyperparameters for the feature map visualization and discussions. For such discussion, the class activity or feature maps of representative examples from the two regions – one in the stable region and the unstable region are shown in Figure 7.2 at some back propagation runs (also known as epochs) of the algorithm. One remarkable information is how the features are able to develop the coherent structures at the 12th layer. The features are also able to visualize shedding of the structures that are not characteristically coherent from the unstable

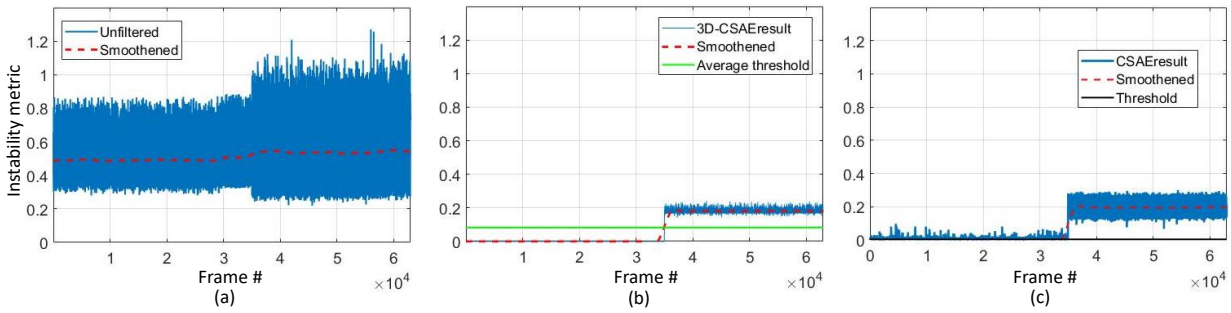


Figure 7.3: Figures showing the performances on a) the training dataset, of b) 3D-CSAE and b) 2D-CSAE with all frames evaluated in the KL-divergence metric, and their respective thresholds.

features while the whole discarded structures are shown in the stable feature maps. Note also that learning proceeds from early epoch (or runs) of the algorithm to the point of minimum noticeable change in the network parameter (i.e, when the training stops). As the algorithm learns better during training, the vortex structure of instability occurring from that of a fully developed coherent structure leads to breaking and shedding of some parts of the structures.

7.3.2 Testing protocols

The main goal of training an intelligent algorithm is to generalize to identifying similar structures in unseen datasets. This is particularly important in this case with the large combination of factors at play and the impossibility of human identification from a single sensor. The testing protocols are set up rather differently from the training protocol because we are not bothered about data augmentation. We simply use the same N value as the training dataset, but do not consider the overlap, k. The assumption is justified by searching for the most general set of parameters for all conditions. We also checked that the frame numbers of prominent intermittent structures in the 2D-CSAE matched those of the 3D-CSAE, and found this to be true for most of the hyper-parameters. Thus, 7 minutes videos at about 3.120 KHz, giving $21,840$ examples are provided in each of the 3 analyzed test protocols. These examples are then converted to $21,840/N$ volumetric examples. Where the frames are end-padded with sufficient number of zero frames if N is not exactly a factor

of 21,840. This ensures that the individual frames can be derived from the volumetric examples. The protocols are identified by the ratio of the air flow (AFR) and the fuel flow (FFR) which are expressed in liter per minutes of the indicated values. Where lpm represents liters per minutes, the protocols are:

1. 500_{40to30}: Protocol has AFR = 500 lpm and FFR is varied from 40 lpm to 30 lpm.

2. 600_{50to35}: Protocol has AFR = 600 lpm and FFR is varied from 50 lpm to 35 lpm.

3. $500to600_{40}$: Protocol has FFR = 40 lpm and AFR is varied from 500 lpm to 600 lpm.

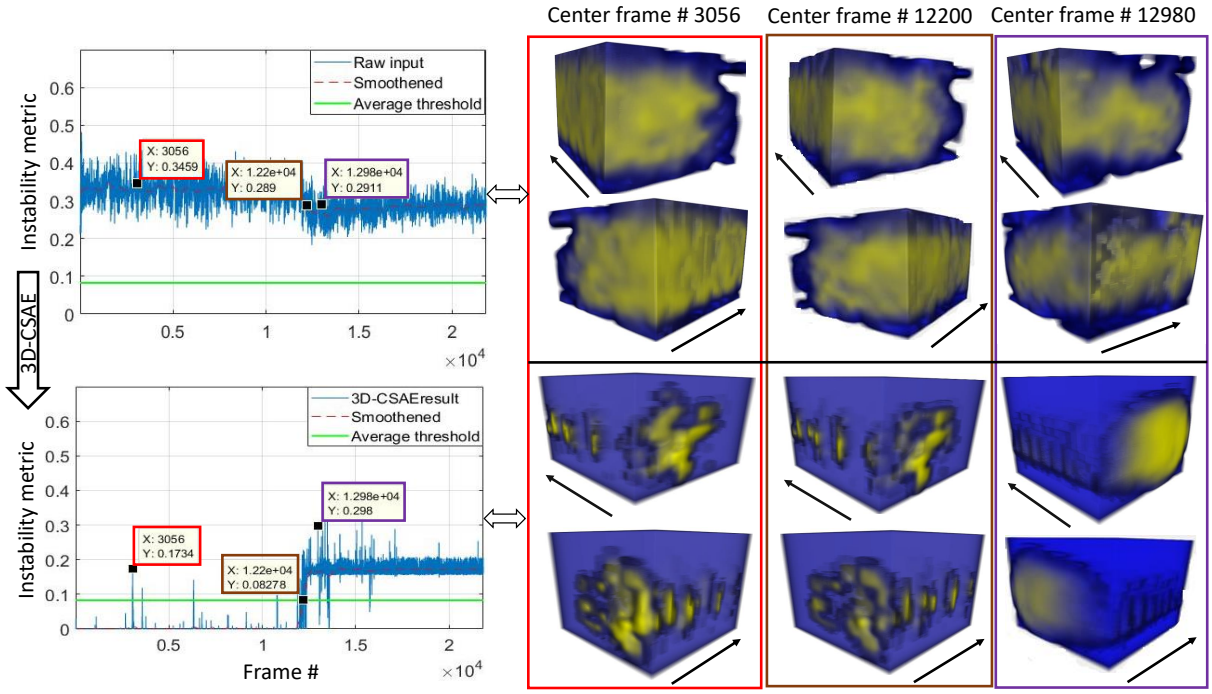


Figure 7.4: Evaluation of the detection results and the visualization of volumetric frames for protocol 500_{40to30}. Arrows show direction of optical stream flow.

The results of the test protocols (Figures 7.4, 7.5 and 7.6) are shown for the overall best hyper-parameter settings. Volumetric visualization show how the filtered and non-filtered frames vary, while the coherent structure are retained. The unfiltered frames are contrasted with the 3D-CSAE results (or the filtered frames). The protocols generally transition from the stable region

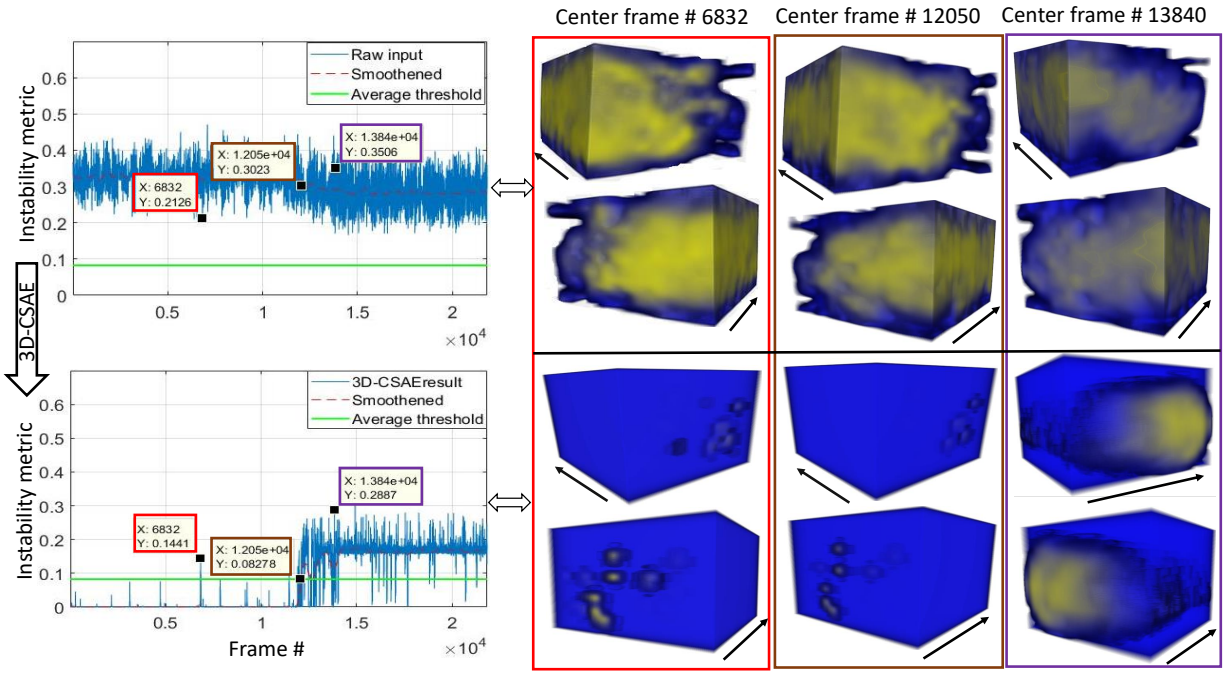


Figure 7.5: Evaluation of the detection results and the visualization of volumetric frames for protocol 600_{50to35}. Arrows show direction of optical stream flow.

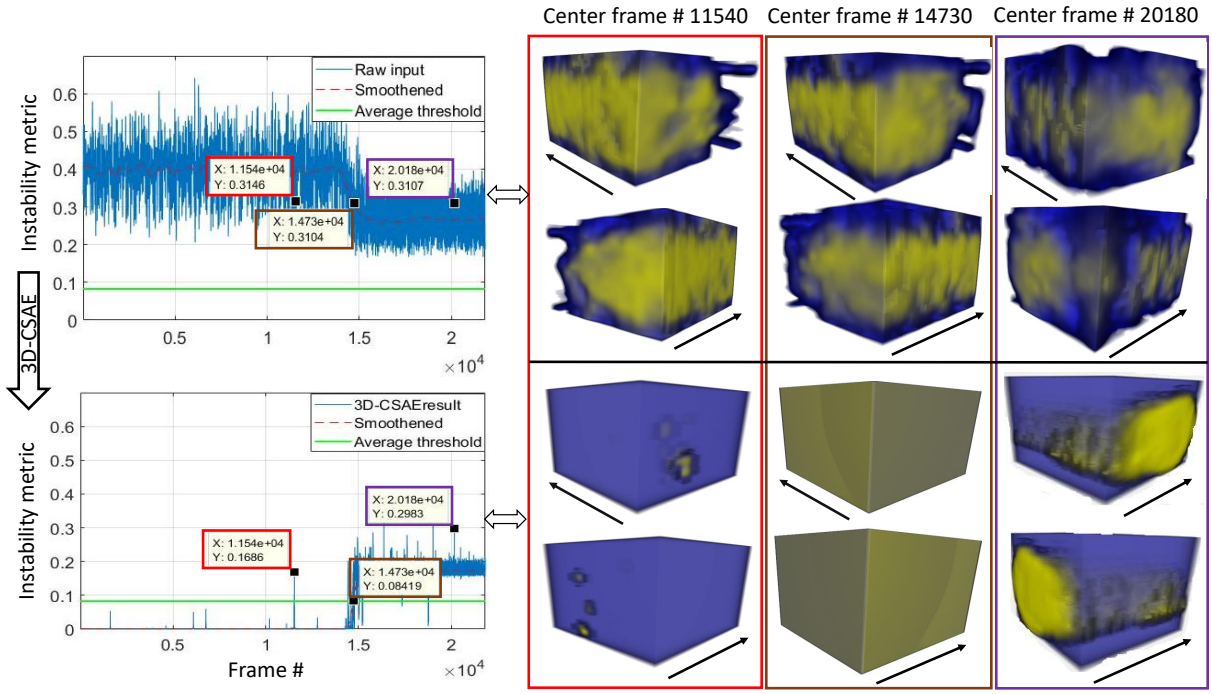


Figure 7.6: Evaluation of the detection results and the visualization of volumetric frames for protocol $500to600_{40}$. Arrows show direction of optical stream flow.

to the unstable region, similar to the 2D-CSAE (Akintayo et al. (2016b)) evaluation, however 3D-CSAE is able to clearly distinguish the stable region from the unstable region, enabling the burst of intermittent structures, which although are similar to the 2D-CSAE results, but show interesting frame-to-frame dynamics (transients) as shown in the frame-by-frame visualization of some examples in Figures 7.7, 7.8 and 7.9.

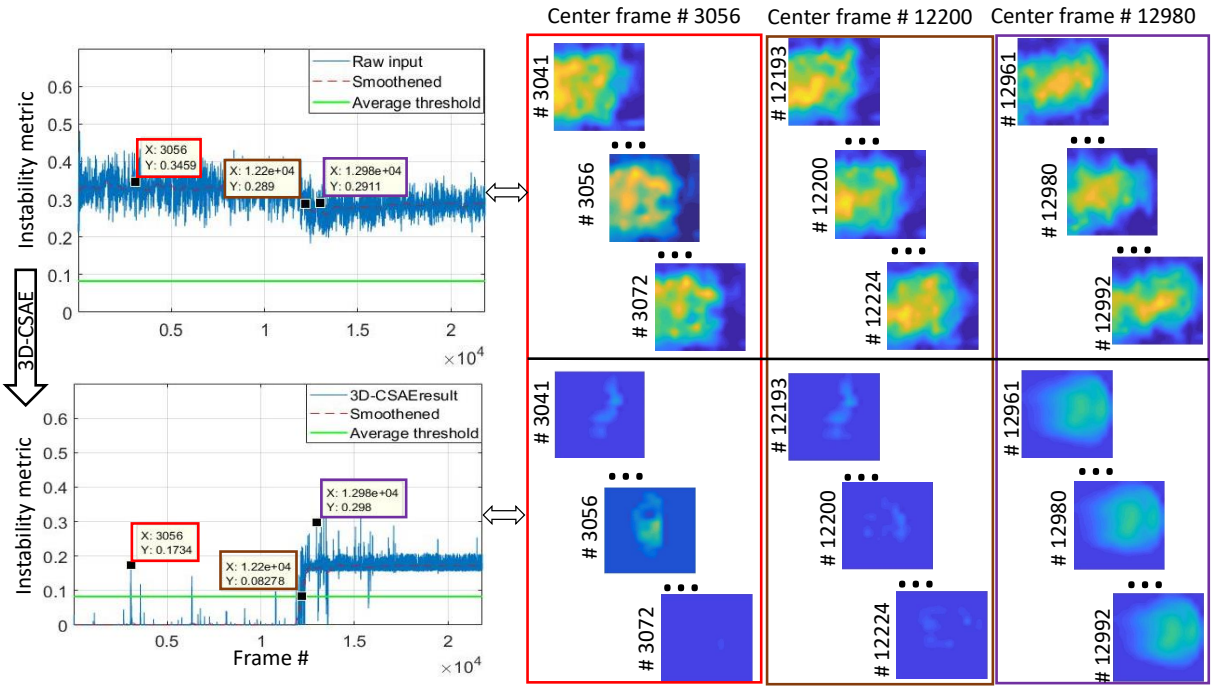


Figure 7.7: Evaluation of the detection results for protocol 500_{40to30} showing the first, identified and final frames of the specially highlighted points.

The frames representing the transition from stable to unstable generally have intermediate properties of all frames in a given volume. The protocol in Figure 7.4 however has more coherent structures than Figure 7.5 and 7.6 in both the stable and transition regions, probably because it has the leanest air flow rate. Thus, the coherent structures exhibit more interesting dynamics, which are meaningful with multiple joint overlapping frames than single frame, where the single shown structure may simply be due to the effect of a single outlier frame. Trends in a volumetric frame of each protocol. The occurrence of structures in the stable frames is consistent in most of the protocols with those of 2D-CSAE, but the evolution around neighboring frames are revealed

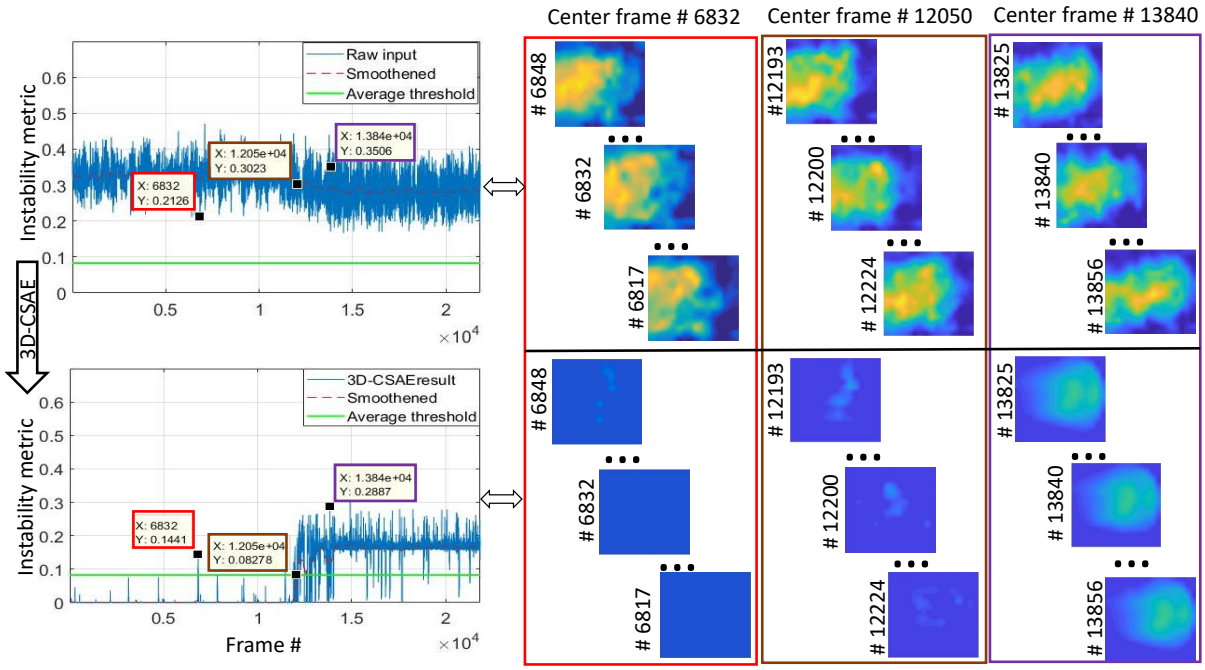


Figure 7.8: Evaluation of the detection results for protocol 600_{50to35} showing the first, identified and final frames of the specially highlighted points.

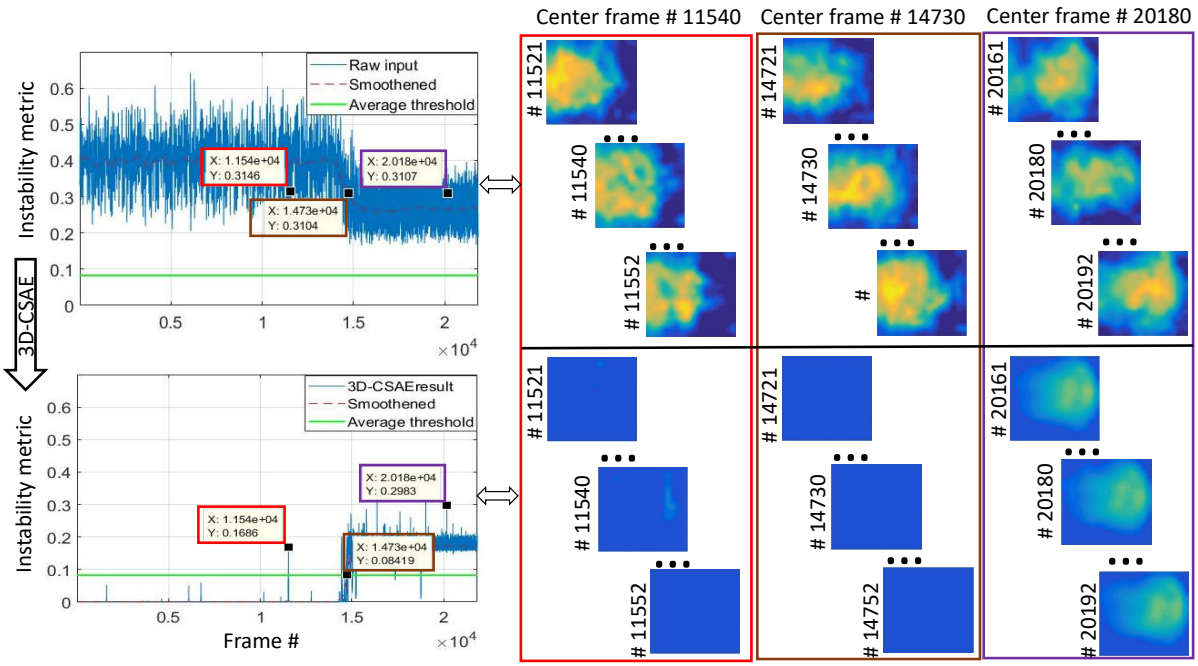


Figure 7.9: Evaluation of the detection results for protocol $500to600_{40}$ showing the first, identified and final frames of the specially highlighted points.

in 3D-CSAE results. The last results confirm the presence of 'combustion noise' when the air flow is increased in Figure 7.6, rather than reducing the fuel flow rate in the Figures 7.4 and 7.5, but 3D-CSAE is still able to resolve the instability region.

7.4 Summary

In summary, the study shows the potential of deep learning techniques 3D-CSAE in characterizing the stability properties of the flame video from a swirl stabilized combustor. The technique significantly improved upon the 2D-CSAE and enables the visualization of the characteristics, and evolution, of interesting dynamics that leads to combustion instability. The temporal streams in addition to the spatial correlation property of convolution are effective for modeling the multi-scale and multilevel correlation present in engines. From a sensing perspective, it is appealing that many interesting properties can be revealed by simply measuring only flame videos, rather than fusing observations (such as pressure, chemiluminescence, velocity, etc.) with increased sensor costs and reduced reliability. Finally, we note that similar analysis can be conducted on observations from real-life engines after some simulations (or training) have been done for all observed operating conditions.

CHAPTER 8. SUMMARY, CONCLUSION AND FUTURE RESEARCH DIRECTION

8.1 Summary and Conclusion

Some of the techniques for extracting meaningful features from diverse dataset types have been implemented in this research. The projects have utilized the features learned from datasets to facilitate transfer of the performance of learned models to unseen datasets. A series of application areas varying from degraded environment through to energy prediction have been implemented. In particular, this research has improved on discriminative models by developing a selectivity technique for discriminability in an unlabeled data setting.

Some of the significant contributions that were included in this research are summarized here. The first contribution was the development and design of a low light network (LLNet) (Lore et al. (2017)) for enhancing images taken in poorly-lit and noisy environment or with cheap imaging sensors. The second and very significant contribution to knowledge was the development of a convolutional selective autoencoder (CSAE) (Akintayo et al. (2016b)) for detecting the onset of combustion instabilities using flame videos collected in a swirl-stabilized combustor. The model was extended to a 3D version which included temporal flow to enhance the performance of the spatial (2D) model. The robustness of the third contribution was demonstrated on the hitherto stubborn application of detecting soybean cyst nematode eggs (SCN) (Akintayo et al. (2016a)) eggs on image frames collected from infested farms after the frames have undergone some laboratory preprocessing. The third contribution dealt with the development of a hierarchical symbolic dynamic filtering (HSDF) (Akintayo and Sarkar (2017)) technique for extracting salient features from time series dataset with application to the publicly available reference energy disaggregation dataset (REDD). The final contribution was the prediction of energy generated and demanded using a spatiotemporal pattern network (STPN) (Jiang et al. (2017b)).

The conclusion here is provided in the light of the spatial, temporal and spatiotemporal datatsets from which the hierarchical features were extracted.

8.1.1 Feature extraction from spatial datasets

Feature extraction from this class of datasets have been examined by adapting fundamental techniques. First, a stacked sparse denoising autoencoders (SSDAs) (Subsection 2.6.1) which was originally designed for image denoising has been adapted in training for enhancing natural images in taken in low light environment by utilizing the hierarchical spatial features in models that result from training on synthetically darkened images. Also, a convolutional autoencoder (CAE) (Subsection 2.6.4) has been improved for selectively training images such that hierarchical features that represent objects (or regions) of interest/disinterest are extracted from the images.

8.1.2 Feature extraction from temporal datasets

A Hierarchical Symbolic Dynamic Filtering (HSDF) modeling of quasi-stationary behaviors in unsupervised, streaming-type time series has been designed as a computationally simple and efficient technique for extracting hierarchical features from slow time-scale non-stationary time series data that comprises of quasi-stationary time series segments. The underlying concepts of probabilistic finite state automata (PFSA), Chinese Restaurant Process (CRP), stickiness and likelihood change rate have been designed into the proposed hierarchical framework. The quasi-stationary dynamics are captured at a fast time-scale using individual PFSA models at a lower layer, transitions of the system at a slow time-scale among different quasi-stationary dynamics are captured using similar PFSA model at a upper layer. It has been showed that tracking the change in likelihoods of different unique quasi-stationary characteristics leads to a more efficient algorithm. The new feature has been incorporated via a novel adaptive CRP formulation. The proposed algorithm is tested and validated to extract features of features from various temporal datasets.

8.1.3 Feature extraction from spatiotemporal datasets

First, a novel data-driven, spatiotemporal pattern network (STPN) has been used to predict energy consumption for both supply side and demand side energy systems. While symbolic dynamic filtering (SDF) (Subsection 2.7.2) performs the discretization and symbolization of continuous domain data for data-level fusion of different variables in a dynamical system, a D-Markov machine is able to capture its temporal characteristics. This work establishes another PFSA, called xD-Markov machine, to uncover the causal dependencies between two time-series in this work. Moreover, for the quantification of causal dependencies, a mutual information based metric is applied. Prediction based on the STPN framework is proposed using expectation from symbolic domain to symbolic and continuous domains. On the other hand, a 3-dimensional convolutional selectional selective autoencoder has been designed for visualizing the volumetric features in hi-speed videos. The videos have been decomposed into short bursts of windowed images for training the 3D-CSAE algorithm.

8.2 Future Research Areas

Leaning on the shoulder of giants in the field, this research has presented significant improvements on the current best feature extraction techniques. However, the concepts utilized are still very far from achieving the main goal of artificial intelligence. Therefore, the following concepts are introduced here as viable next step, taking off from the current research proposal.

8.2.1 Generative adversarial networks (GAN)

Generative models can be classified as stochastic and non-stochastic depending on the manner in which the sample generating model is sampled from the layer above the current one. The major non-stochastic class of generative models discovered are the auto-encoder networks (Vincent et al. (2008); Rifai et al. (2011)) and the noise contrastive estimation (NCE) (Gutmann and Hyvarinen (2010)). On the other hand, the stochastic generators have been given much attention, and are further divided into:

- the undirected graphical models with examples such as deep Boltzmann machines (Salakhutdinov and Hinton (2009)) and generative stochastic networks (Bengio et al. (2014)), and
- the directed graphical models having the neural autoregressive distribution estimator (Larochelle and Murray (2011)) as example.

The directed graphical examples have a major dis-benefit of being sequentially executed (not parallelizable), hence they are not included in this study. The last category are the mixed stochastic and non-stochastic are the most recent generative adversarial network (Goodfellow et al. (2014)) and deep convolutional generative adversarial network (Radford et al. (2016)). A combination of recent skills for reducing the instability effects previously noticed on such networks. The more important contributions with GANs are:

- input is the noise distribution, z, and the true data distribution, x
- rectified linear units were the activation for the generative layers
- maxout activations were used in the discriminative layers
- dropout in discriminative layer

Generative Adversarial Network (Goodfellow et al. (2014)) improves the NCE by employing the discriminative model, D to monitor the generative model, G that has been adapted to a discriminative training condition. D is trained to maximize its discriminative ability of the true distribution and generative model's distribution and train G to minimize its generated distribution from the true data distribution, i.e., G to maximize the mistake of D from the data distribution. A sequential form the min-max functional:

$$\max_{D} \min_{G} \Big(\mathbb{E}_{x \sim P_{data(x)}} log(D(x)) - \mathbb{E}_{z \sim P_{data(z)}} log(1 - D(G(z))) \Big) \tag{8.1}$$

A deterministic training of GAN can overcome the need for variational inference that is obtainable in stochastic type network.

8.2.2 Recurrent neural networks (RNNs) and long short term memory networks (LSTMs)

In the temporal feature extraction domain, although short-ranged dependencies can still be captured by the feed-forward deep network as was the case in Akintayo et al. (2016b) where convolutional selective autoencoder was implemented, a parallelized, distributed technique is required for handling time sequence of the video. Recurrent neural networks and long short term memory networks and their variants would probably be the most suitable for the problem. Recurrent neural networks (RNNs) (Graves and Schmidhuber (2005); Pascanu et al. (2013); Graves (2014)) model sequence-type networks, especially when problems like image captioning, video classification and machine translation by components that incorporate memory in the model. The major architectural addition for achieving these goals is the feedback loop for the network state. In the feedback process, the network is given some memory for managing what information is retained at the output, unlike the standard feed-forward types so far considered.

Long short-term memory, LSTM (Gere et al. (2000)) enables deep-stacking of recurrent neural networks by reducing the consequent effect of non-uniform gradient through in the layers of the network that arises. LSTMs enable recurrent networks to be trained via backpropagation through time, BPTT (Mozer (1989)) by unrolling the state dynamic feedback loop as 2 feedforward loops whose weights are shared usually by averaging. LSTMs model the state machines with components such as the blocks, units and gates mainly for operating on the input sequence. However, the problem with training such structured network has been discussed (Pascanu et al. (2013)) to be amongst others, the presence of many several local minima and saddle points. Also, there are certainly more weight symmetricity that have the potentials of resulting in increased non-identifiability problems over the feedforward versions.

APPENDIX A. ARCHITECTURAL PLATFORM AND HARDWARE

A.1 Introduction

Before the proliferation of high performance computing devices, processing times with Central Processing Units (CPUs) have been reported to increase exponentially with number of feature (Lee and Landgrebe (1993)). Nowadays, such computation times for extracting similar features have drastically reduced by several hours using graphics card that have several CUDA cores (Mckee (2017)). The main trade-off between CPUs and GPUs are latency and computation throughput requirement. While GPUs provide better throughput, latency for accessing the cache still favor the CPUs. GPUs also improved the energy efficiency by their increased silicon use. The important factors in choosing such hardware are: the model number, memory bandwidth that is related to the Dynamic Random Access Memory (DRAM), the ease of 'crossfire or 'link multiple graphics cards, the host (or CPU) efficiency, the wattage rating, its number of connectors, the size of the computation to be done, etc. Nvidia® corporation is a reputed company in the history of graphics card Nvidia ® launched its compute graphics card in 2006. The company was known as the first to adapt GPUs for computation. The main idea with GPUs are to utilize many more lower frequency and less energy-consuming cores than those of the CPUs for scanning and performing computations on units of higher dimensional arrays in parallel. The tasks to be performed by the block of threads are called the jobs. Sometimes, the jobs are self-containing (i.e., computation is completed without needing the host CPU), heterogeneous computing involving both GPUs and CPUs have been leveraged. The latter allows slower but large memory requiring computation while the former does its job of fast distributed computing. Some examples of syntax are the register declaration with a type, arithmetic and logics operation, memory type declaration and complex operations such as branch. Previously slow matrix decomposition computation were among the first algorithm to benefit from the compute capability of GPUs (Davis (2006)). The procedure

for linking remotely to the compiled program at runtime is the Single Program Multiple Data (SPMD). The compilation of codes in CPUs are usually interpreted with C/ C++ compilers while GPUs use the custom written compilers and assemblers such as NVCC or Conpute Universal Device Architecture (CUDA). For instance, the main steps in parallelizing each CUDA files are shown in Figure A.1. Several drivers have facilitated the graphics processing objectives such as shaders

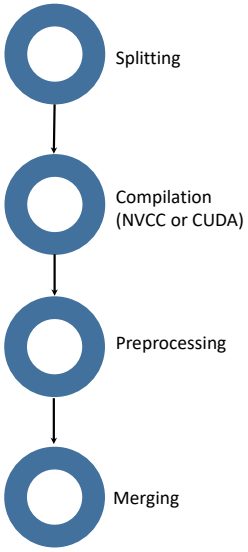


Figure A.1: Steps in CUDA computings

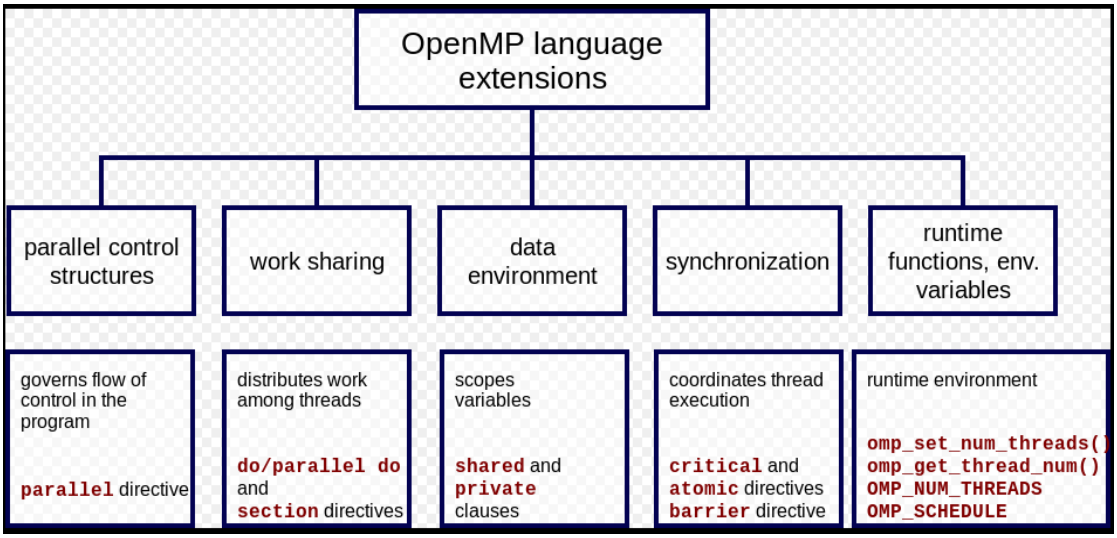
as well as high performance computation such as efficient lower-upper matrix decomposition. The compute universal device architecture (CUDA) driver, DirectCompute by Microsoft® and OpenCL by Khronos/Apple ® are examples of such drivers.

A.2 GPU Computing Libraries

There are increasing number of computing software that efficiently utilize the heterogeneous and hierarchical computing capabilities of GPUs. A quick search of deep learning software will reveal that there are quite a few of them already available. This section will describe few of useful drivers that enables training deep learning software.

A.2.1 Open Multi-Processing (OpenMP) and Open Computing Library (OpenCL)

Shared memory multiprocessing (OpenMP) is a transparent (in terms of sharing) parallelization library where a master thread engages a number of slave threads, which operates either in parallel or in series. Each slave thread has an ID which is spurned to enable performing part of the code. Runtime allocates threads to differe==nt processors. Performance of OpenMP is dependent on vendor and a set of clauses usually included as functions to enable the application programming interface (API) – OpenMP to perform the tasks shown in Figure A.2 (Commons (2017)). For instance, parallel control structures are in place to govern the structure when parallel computation is done. Despite the advantage, memory bandwidth limitation, race conditions and hierarchical sequential execution waiting reduces the effectiveness at most times.



Courtesy: "https://commons.wikimedia.org/w/index.php?title=File:OpenMP_language_extensions.svg&oldid=246140665"

Figure A.2: OpenMP extension language schematic.

In addition to OpenMP's parallelizing capability, heterogeneous computing interfaces GPUs with CPUs or other devices such as digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGAs) are enabled via Open computing language (OpenCL). OpenCL was reported to be originally developed by Khronos research group at Apple®, but now has the support of other

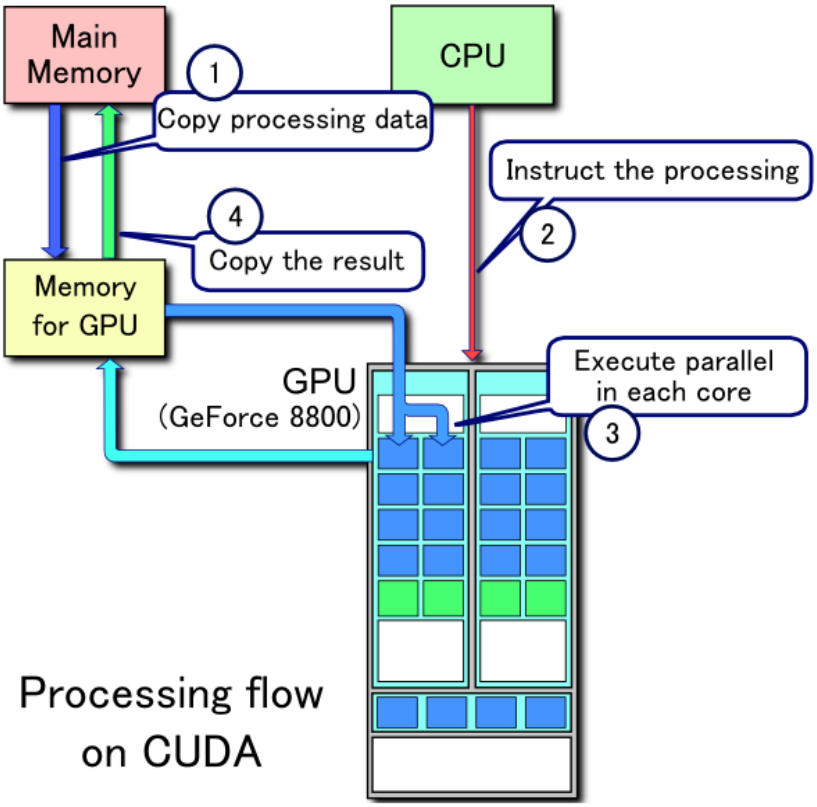
companies and it is also open to the community for continuous development. OpenCL can spurn several computing units from a single computing device ensures that languages for OpenCL are compiled at runtime. The hierarchy of memory in OpenCL ensures that there is global, low latency read-only memory, local memory and private memory. The API has its own version of C language, termed OpenCL C.

A.2.2 Compute Universal Device Architecture (CUDA)

CUDA (Mivule et al. (2014)) was developed by Nvidia® as an improvement on OpenGL and Direct3D for easier parallel programming. It offers real time high resolution 3D intensive computation plus multi-core applications. CUDA is touted to have OpenCL capability, therefore uses its own Cuda C programming that could be written in C, C++ or Fortran. The data goes through from the main memory to the GPU memory and back while virtual instruction sets flow from the CPU to the GPU as shown in Figure A.3, courtesy of Commons (2017). The instructions are either MIMD or SIMD depending on the parallelization and heterogeneity provided. Since they are operated on multiple grids which have multiple threads and ultimately multiple processing elements called cores (Figure A.5). This simple flow enables complex computations such as fast Fourier transform computation, large number sorting efficiently. CUDA has versions which improves specified libraries with added capabilities such as CUDNN – the CUDA deep neural network for efficient deep network models, CUSOLVER – the dense and sparse matrix solver of CUDA, etc. CUDA also enables 3D point cloud processing, complex graph theoretic analysis such as push-relabel algorithm as well as sorting of large sequence in addition to efficient matrix multiplication.

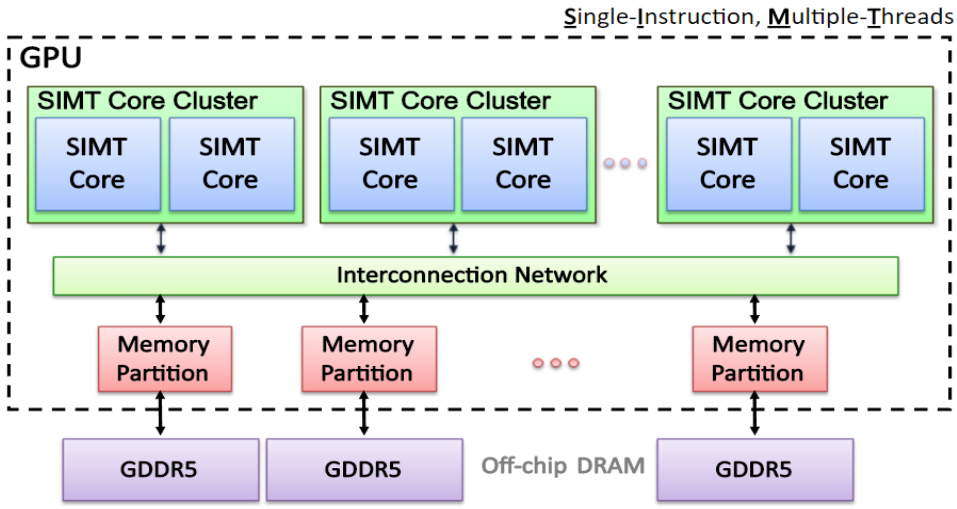
Thread execution are an important factor of the architecture by the manufacturer. Nvidia Corporation for instance uses the virtual type parallel thread execution (PTX) in the system (Figure A.4), while organizations such as AMD, Samsung etc., designed the heterogeneous system architecture (HSA) type of Virtual instruction set architecture.

The multiprocessor of graphics processing unit with the host interface is shown in Figure A.4. Single program multiple data (SPMD) utilizes multiple threads, the threads in turn have several



Courtesy: Wikipedia

Figure A.3: Schematics of the CUDA processing flow with GeForce 8800 illustrative GPU.

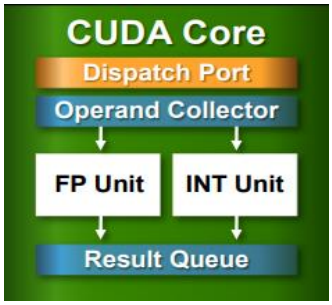


Courtesy: "http://slideplayer.com/slide/9139861/"

Figure A.4: GPU Architecture Overview showing the instruction multiprocessing threads.

cores which are grouped together as clusters. An interconnection network splits and merges the data according to the memory requirement from the host (CPUs) dynamic random access memory (DRAM) to the GPU and vice versa. NVIDIA® GPUs for instance have different micro-architecture that differ in the features. There are also variant of the architecture called the Multiple Instruction Multiple Data (MIMD) where the data and the instructions are both varied. Tesla, Fermi and Kepler microarchitecture are some of the designs by NVIDIA® Inc. for graphics and computation. The CUDA core for a streaming Fermi micro-architecture is shown in Figure A.5

Each core executes parallel threads, while the streaming microprocessor executes a group of threads called blocks. The top level kernel(s) (also the GPU function(s)) execute(s) on the grid (or GPU node). There are also registers and local memory associated with the threads, while the memories are shared by the threads in the blocks, and a higher latency and lower bandwidth (than shared memory) is accessible to all the grids. In summary, while CPUs process data in series, the GPU performs a massive parallel computation by executing kernels as grids of blocks of threads.



Courtesy:
"https://www.cc.gatech.edu/~vetter/keeneland/
tutorial-2011-04-14/02-cuda-overview.pdf"

Figure A.5: Fermi streaming microprocessor CUDA core with FP floating point and INT integer computed in different streams.

A.3 GPU Hardware

The hardware support for the computations are becoming pretty cheaper and more efficient for the computation. Improvement in designs of the GPUs, low energy consumption and high silicon usage are some of the basic factors that enhance GPU performance. The graphic cards are built on a support base machines, and these base machines are usually cheaper. As at the time of compiling this research, the Self-aware Complex Systems (SCS) laboratory is endowed with the following graphics cards and machines.

GeForce Titan Blacks: Titan blacks were released by Nvidia®s in 2014. The cards are fast, powerful, cool and quiet. Titan black has 2880 CUDA cores by design. They have 6GB of Virtual Random Access Memories (VRAMs). The cards were Nvidias first success at optimizing both gaming and computation experience. However, like the recent improvements, Titan Black utilize the PCI express cards 3 for improved communication bandwidth. Titan Blacks support HDMI, VGAs and have a video. Including specifications by Nvidia (2017), they have a video. As of now, Titan black cards only support CUDA with compute capability of 3.0.

Geforce Titan X: GeForce Titan X was introduced in the early part of 2016, and it was based on the Maxwell type streaming multiprocessor. Titan X (Maxwell) has pretty good single floating point performance, sometimes similar to the more recent versions, P40s. Titan X has 3072 CUDA cores that ensure time-effective computation. The cards have 12GB of Virtual Random Access

Memories (VRAMs), Titan X is very efficient in price ($1000) of Floating Point Operations per seconds (FLOPS) at 6.9GFLOPS/$) compared to the recent P40s (1.7GFLOPS/$). Titan Xs and P40s are enabled by the current latest CUDA compute capability (of 5.2).

Pascal Titan X: Titan X (Pascal) provides some significant improvement on the performance of the Titan X (Maxwell). The cards were released in the middle of 2016. Some of the improvements to the Maxwell version include 3584 CUDA cores, faster clock and memory application rates.

Pascal P40: Pascal P40 was released by Nvidia® in late 2016. Cards have pretty good double floating point performance than the Titan X and include INT8 capability for inference. The cards have 3840 NVIDIA® CUDA cores. With 24GB of VRAMs, Pascal P40s are suitable for high dimensional computation such as 3D point cloud reconstruction. Pascal P40 cards utilize air cooling in multiple directions as the trade-off for the improved performance. They have a compute capability of 5.2 for CUDA computation. However, they are more expensive at $3000 per card (Redmon (2017)). The latest so far are the Tesla K80 which are $5000 per card.

Among the several more graphics cards support in SCS laboratory, the cards in Table A.1 have been particularly useful throughout the period of this research.

Table A.1: Available facilities in Self-aware Complex Systems laboratory in September 2017

Graphic card	Quantity (pieces)
Nvidia® GeForce Titan Black	2
Nvidia® GeForce Titan X	4
Nvidia® Pascal Titan X	4
Nvidia® Pascal P40	4

Currently,The Titan Blacks operate via Nvidia®'s runtime driver version 375.66, the GeForce Titan Xs operate via the driver version 385.66 while the Pascal Titan Xs and the P40s utilize driver versions 375.26, but these could be easily updated. This research enjoyed access to all the machines and the clusters, but the primary machine had one Titan black and a Titan X (Maxwell).

A.4 Summary

In summary, decisions on speed and performance of a machines are dependent on the available capital resources. However, the type of project deliveries should also be considered before deciding for particular hardware.

APPENDIX B. ALGORITHM IMPLEMENTATION SOFTWARE

Here, we provide a brief discussion of the popular deep learning software that is available for the massive parallel and heterogeneous computations required by machine and deep learning code.

B.1 Trending Deep Learning Platforms

We illustrate all the available software with an example of defining the simple logistic regression layer. Mathematically logistic regression is defined for a given class Y=1 given a unit i of the input x(i) and all units indexed by 1, $\cdots$ j, $\cdots$, J as, $P(Y = 1|x(i)) = P^I = \frac{e^{Wx(i)+b}}{\sum_{j=1}^{J} e^{Wx(j)+b}}$ where W and b are the weights and biases of the network linking the input units to the output units. Generally, the platforms are developed and managed by industries such as Google Inc. (TensorFlow), academic groups such as Universite de Montreal (Theano) as well as by individual efforts such as Francois Chollet (Keras). In these computing software, the important considerations are the details on the operating systems platforms supported, the backends of the software that use the computation graphs, the interfaces of the software and other parallelization (OpenMP or CUDA), computing library (OpenCL) and models supported.

B.1.1 Caffe

Caffe was created by Yangqing Jia and developed by Evan Shelhamer (Jia et al. (2014)) at University of California, Berkeleys Vision and Learning Center (BVLC). Caffe is built directly on C++ backend but can be wrapped with Matlab (MatCaffe) and Python (PyCaffe) front end. It has active support of OpenMP. However, in the original form, OpenCL support was not provided, but under the hood synchronization of the data in CPUs and GPUs as blobs are possible. It is arguably the fastest parallel platform because of its text-forms natural independence of front end programming language. Also it has the capability of allocating memory on demand. Caffe has the

shortcoming that the definition of the text-like model usually involves more explicit coding skills. For example, a Logistic regression would be defined thus:

name: "log-reg",

type: "Data",

top: "data",

top: "label",

data param { { { **source**: "Input leveldb" **batch size**: } } } ,

Algorithm 4: Caffe Logistic regression definition

The example layer (logistic regression) and the data parameters (source and batch sizes) are differently defined and usually require using more lines of code to achieve simple tasks. The top and the bottom type data definition requires understanding the connectivity of the computation graph. Also Caffe has no support for the generating model class of Restricted Boltzmann Machines (RBMs). The latest version called Caffe 2 has been recognized by Facebook and Yahoo, among the popular big promoters.

B.1.2 Theano

Theano was developed by a team of researchers at LISA Lab at Universite De Montreal, Canada (Bergstra et al. (2010b); Team et al. (2016)), sometimes in 2008. It was built on Python but has cross OS platform support. However, the Python backend usually still has C++ support for some libraries such as the pooling layer. It uses the current symbolic operations for defining the graph of computation, layers, network and optimization. One advantage of its development is the support it provides for other wrappers such as Nolearn, Lasagne, Keras, which ease layer definitions in Theano. It is still touted to be amongst the most effective deep learning tool in terms of computation accuracy. Its layers are first defined, (being symbolic) but the datasets are supplied at compile time. These definitions are also transparently revealed in the codes. Theano tensors are a very important part of the symbolic definition, although visualizing the tensors for debugging may require some extra effort of using the *get_value()* function. So also, they may be set with

the *set_value()* function. Theano supports the primitive deep learning models such as restricted Boltzmann machines and recent reinforcement learning models. Convolutional networks developed by LeCun et al. (1998a), termed LeNet, was among the models that boosted the widespread use of Theano. An example logistic regression layer would be defined as follows:

import LogisticRegression, theano.tensor $\cdots$

LogisticRegression(input, n_in, n_out)

Algorithm 5: Theano Logistic regression definition

where n_in and n_out are the input and output number of units to the layer. A downside of Theano is that its networks' (and layers') definitions are not sequential and hard to think about, but could be interesting once understood. Theano is also hard to debug, and it throws many confusing errors with even the slightest wrong definition of the network. The latest Theano version available in *PyPI* is 0.10.

B.1.3 TensorFlow

TensorFlow was a deliberate effort by Google to develop a faster software version of an older machine learning software DistBelief. TensorFlow (Abadi et al. (2016)) was developed on Python, C++ and CUDA to operate on multiple operating systems. TensorFlow encourages heterogeneous computation via multi-GPU processing. This is facilitated by the common collectives offered by NCCL (pronounced as Nickel) driver that enables reduction in the associated communication bandwidth. TensorFlow provides multiple capability for a wide range of tasks. It specifically aided the best performing speech recognition model. Tensorflow also runs on mobile Android and iOS platforms. TensorFlow's goal was previously for high throughput, lower precision evaluation which was not suitable in model training. However, the requirement has improved for better accuracy in recent years. TensorFlow has one of the coolest user interface called Tensorboard, which runs on a local server of the machine. It can be used to monitor training, validation and testing progress of the algorithm remotely, to visualize the network graph and flow of the data through the network, among other things. A neural network model in TensorFlow starts by first inserting a placeholder

for input tensors. Tensorflow has a very good documentation and allows several configurations to be set in the algorithm. A logistic regression layer in TensorFlow can be added to a *TensorFlow.Session()* as,

import tensorflow

...

tensorflow.nn.softmax(dim, Name)

...

Algorithm 6: TensorFlow Logistic regression definition

where dim is the number of output units and Name is the layer name. TensorFlow has about the largest number of promoter companies, among which are Dropbox Inc., QUALCOMM, Intel, AIRBUS, Airbnb, etc. The latest stable release of TensorFlow 1.3.0 was released in September 2017.

B.1.4 Torch

Torch (Collobert et al. (2011)) was developed by Ronan Collobert, Koray Kavukcuoglu and Clement Farabet (initially released in 2002) to run on LuaJIT (Lua Just-In-Time) which is also built on C programming language. Torch bears close similarity to MatLabs object oriented definition, thus it is very intuitive. Torchs most important advantage is that it can be applicable to all tasks (e.g., computer vision, natural language processing, speech and text processing), it has support for almost all the APIs needed and has libraries for current and earlier models. Torch has an improved training speed because of its design to use the CUDA driver is significantly encouraging. It has the capability for GPU training with double tensor (or float 64), enabling better accuracy. Torch also permits multi-GPU computation. Torch has been extended for use on mobile operating systems. It has been used to build hardware implementations for data flows like those found in neural networks. A logistic regression inherits some inputs from the previous layer and just accepts

the number of output units. A neural network model simply adds,

. . . ,

require nn,

model.add(nn.LogSoftMax(int n_units)),

. . . ,

Algorithm 7: Torch Logistic regression definition

The main caution with using Torch found in forums relates to knowing when to convert the data input to CUDA tensor. Facebook Inc., Google Inc., IBM, Idiap Laboratories are among the promoters of Torch software. Torch 7 is the latest version of the software. Torch became a competitor with TensorFlow with the introduction of PyTorch. PyTorch enhances Torch in capabilities such as the speed, data loading and graph creation. However TensorFlow is known to trump PyTorch in data serialization to other software, deployment on hardware platform and device management, among many others.

B.1.5 Keras

Of the most recent deep learning software friendly front end interface, Keras stands out because of its active development. It was originated by Francois Chollet but is now open source. Keras capability is increasing day by day because it easily wraps other deep learning platforms such as MXNet, Deeplearning4j, CNTK, Chainer, TensorFlow and Theano. Keras was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Keras has the advantage of easy, intuitive, platform-independent (mostly) layer definitions and usually inherits the qualities of the backend. A logistic regression layer is added in this library as follows,

import Keras

. . .

model.add(Dense(output_dim, input_dim, activation='softmax'))

. . .

Algorithm 8: Keras Logistic regression definition

where dim represents dimensions. The main disadvantage of Keras found so far is that certain distributed models such as denoising autoencoders and generative adversarial networks are easier defined in the lower level software due to the top level coding that Keras offers. Also, some APIs might not be available for some backend while they are available for other back ends. From practice, getting 3D features in TensorFlow serving may be hard to work around (or might not yet be available). Keras version 2.0.3 is about the latest version of the software.

B.2 Data and Model Parallelism in GPUs

A bid to reduce the computational time required for training a deep network with the limitation of graphics cards capability is to use multi-GPUs similar to using multi-cores of CPUs for processing datasets in parallel rather than sequentially. A cnn for instance require doing same computation over several individual (or batches of) images. The computation time required also includes waiting time before the sequential execution. However, by simply running the same program on several data in parallel, the computation time can be reduced effectively. The second problem is the transfer of data between the multiple GPUs for common computation at each iteration. The trade-off between the single GPU and multi-GPUs are the dataset throughput speed and the result convergence. Note: The assumption is that the processes are independent and can run concurrently. Training using parallelism may be asynchronous or synchronous. While the asynchronous training techniques enabled distributing the parameters, where such parameters usually are stale and may hurt convergence. Collecting the hyper-parameters from multi-GPU synchronously has the potential to improve the efficiency. In synchronous data parallelism the workers and the master perform the same computing and collective communication tasks at the same time with respect to some shared variables.

Multi-GPU and multi-node uses communication collectives library (i.e., NCCL) functions such as all-gather, all-reduce, broadcast, reduce, reduce-scatter and automatic detection of network topology to determine the optimal communication path. This parallelism library uses the CUDA-aware message passing interface (called) for shared multiprocessing as well as the NCCL for

enhancing communication. The convergence and capability of the MPI-based framework were described by Ma et al. (2016). The package enabled using multiple GPUs with Theano in the case of data parallelism, implying that whenever input data can be divided up among workers and the final results returned to the host in a synchronous way. The goal is to make as little deviation from using Theano with single GPU, running the same program on data batch and average parameters after each iteration. The point-to-point (P2P) transfer and collectives operations are facilitated by the MPI CUDA-aware GPUDirect P2P technology.GPUs already have interconnect capabilities through PCIe and NVLink, so CUDA-awareness can enable the MPI operations to be GPU-based. An all-to-all-sum-all-gather can then be used for the computation and communication. The data precision would be reduced for data transfer and restored for data computation to reduce overhead. The Theano-MPI (Ma et al. (2016)) was employed for Bulk Synchronous Parallel, Elastic Averaging SGD (EASGD). CUDA-aware communication has the advantage that all GPUs can communicate without sending parameter to the host. Platoon is another data parallelism platform that provides asynchronous data parallelism within the compute nodes. Platoon however requires the *Pycuda* and *zeromq* libraries

B.3 Summary

In summary, the basic requirement for all the software platforms are defined as: What shape should the input data be, and what file formats are acceptable? How are layers and data defined for the network? How modular is the software for defining new layers? How accessible are the back end libraries for transparent computation and code debugging? How are the hyper parameters modified for the experimentation? What form, shape and format should the network results and learned model take? How easily can the results visualized during computation? The most fundamental question for electing to use any platform is if there are already developed working model that achieves my task or some other similar tasks. The key idea is to ensure that one is not re-inventing the wheel, but to keep improving on them.

BIBLIOGRAPHY

(2016). The connectionist models of cognition homepage. http://staff.itee.uq.edu.au/janetw/cmc/chapters/BackProp/index2.html.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning.

Agostinelli, F., Anderson, M. R., and Lee, H. (2013). Adaptive multi-column deep neural networks with application to robust image denoising. In *Advances in Neural Information Processing Systems*, pages 1493–1501.

Agrawal, A., Raskar, R., Nayar, S. K., and Li, Y. (2005). Removing photography artifacts using gradient projection and flash-exposure sampling. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 828–835. ACM.

Aharon, M., Elad, M., and Bruckstein, A. (2006). An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322.

Akintayo, A., Guo, H., Vaswani, N., and Sarkar, S. (2015). Literature review: Low-light images & videos, noise types and denoising algorithms. Unpublished, Iowa State University.

Akintayo, A., Lee, N., Chawla, V., Mullaney, M., Marett, C., Singh, A., Singh, A., Tylka, G., Ganapathysubramanian, B., and Sarkar, S. (2016a). An end-to-end convolutional selective autoencoder approach to soybean cyst nematode eggs detection. *Knowledge Discovery and Data Mining workshop on Data Science for Food, Energy and Water*, pages 1–8.

Akintayo, A., Lore, K. G., Sarkar, S., and Sarkar, S. (2016b). Prognostics of combustion instabilities from hi-speed flame video using a deep convolutional selective autoencoder. *International Journal of Prognostics and Health Management (special issue)*, 7(023):1–14.

Akintayo, A. and Sarkar, S. (2015). A symbolic dynamic filtering approach to unsupervised hierarchical feature extraction from time-series data. *ACC proceedings, Chicago'15*, pages 1–8.

Akintayo, A. and Sarkar, S. (2017). Hierarchical symbolic dynamic filtering of streaming nonstationary time series data. *arXiv:1702.01811v1*.

Aldous, D. J. (1985). *Exchangeability and related topics in Lecture Notes in Mathematics*, volume 1117. Springer Berlin Heidelberg.

Alessandrini, S., Delle Monache, L., Sperati, S., and Cervone, G. (2015). An analog ensemble for short-term probabilistic solar power forecast. *Applied Energy*, 157:95–110.

Barbu, A. (2009). Learning real-time mrf inference for image denoising. *IEEEXplore*, 978-1-4244-3991-1(09):1574–1581.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Batra, N., Kelly, J., Parson, O., Dutta, H., Knottenbelt, W., Rogers, A., Singh, A., and Srivastava, M. (2014). Nilmtk: An open source toolkit for non-intrusive load monitoring. *5th International Conference on Future Energy Systems(ACM e-Energy), Cambridge UK*, pages 1–14.

Beal, M. J., Ghahramani, Z., and Rasmussen, C. E. (2014). The infinite hidden markov model.

Beckel, C., Kleiminger, W., Cicchetti, R., Staake, T., and Santini, S. (2014). The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 80–89. ACM.

Bengio, Y. (2009). Learning deep architecture for ai. *Foundations and Trends in Machine Learning*, pages 1–71.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19(1).

Bengio, Y. and Olivier, D. (2011). On the expressive power of deep architectures. *Algorithmic Learning Theory. Springer Berlin/Heidelberg.*

Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. *Journal of Machine Learning Research*, 32:1–9.

Bengio, Y., Yan, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. *Neural Information Processing Systems*, pages 1–9.

Berger, A. L., Pietra, S. A. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Association for Computational Linguistics*, 22(1):1 – 36.

Bergland, N. and Gentz, B. (2002). Pathwise description of dynamic pitchfork bifurcations with additive noise. *Probab. Theory Related Fields*, (122):341–388.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010a). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy).* Oral Presentation.

Bergstra, J., Breulex, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010b). Theano: a cpu and gpu math expression compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy).* Oral Presentation.

Berkooz, G., Holmes, P., and Lumley, J. L. (1993). The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25(1):539–575.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer Science-Business Media LLC, Microsoft Research Limited, Cambridge CB3 0FB, U.K.

Blei, D. M. and Jordan, M. I. (2006). Variational inference for dirichlet process mixtures. *International Society for Bayesian Analysis*, 1(1):121 –144.

Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. *Neural Information Processing Systems*, pages 1–8.

Brock, W. A. and Carpenter, S. R. (2006). A leading indicator of ecological transition. *Ecol Letter*, 9(3):311–318.

Bryd, D. W. J., Barker, K. R., Ferris, H., Nusbaum, C. J., Griffin, E. W., Small, R. H., and Stone, C. A. (1976). Two semi-automatic elutriators for extracting nematodes and certain fungi from soil. *Journal of Nematology*, 8:411–413.

Burger, H. C., Schuler, C. J., and Harmeling, S. (2012). Image denoising: Can plain neural networks compete with bm3d? In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2392–2399. IEEE.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2.

Candes, E. J., Li, X., and Soltanolkotabi, M. (2015). Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007.

Chakraborty, S., Sarkar, S., Gupta, S., and Ray, A. (2008b). Damage monitoring of refractory wall in a generic entrained-bed slagging gasification system. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 222(8):791–807.

Chakraborty, S., Sarkar, S., Gupta, S., and Ray, A. (October 2008a). Damage monitoring of refractory wall in a generic entrained-bed slagging gasification system. *Proceedings of the I Mech E Part A: Journal of Power and Energy*, 222, Part A(8):791–807.

Chan, R. H., Ho, C.-W., and Nikolova, M. (2005). Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization. *Image Processing, IEEE Transactions on*, 14(10):1479–1485.

Chatterjee, P., Joshi, N., Kang, S. B., and Matsushita, Y. (2011). Noise suppression in low-light images through joint denoising and demosaicing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 321–328. IEEE.

Chattopadhyay, I. (25 Jun, 2014). Causality network. *arXiv:1406.6651v1[cs.LG]*.

Chen, S. and Wang, Y. (2013). Convolutional neural networks and convex optimization. Report, University of California San Diego, Department of Electrical and Computer Engineering.

Chen, T., Ma, K.-K., and Chen, L.-H. (1999). Tri-state median filter for image denoising. *Image Processing, IEEE Transactions on*, 8(12):1834–1838.

Cheng, H. and Shi, X. (2004). A simple and effective histogram equalization approach to image enhancement. *Digital Signal Processing*, 14(2):158–170.

Chollet, F. (2015). Keras. GitHub: https://github.com/fchollet/keras.

Coen, M. (2005). Cross-modal clustering. *In Proceedings of the 12th National Conference on Artificial Intelligence(AAAI'05)*, pages 932–937.

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. *NIPS workshop*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing:deep neural networks with multitask learning. *25th International Conference on Machine Learning*, pages 1–7.

Cominola, A., Giuliani, M., Piga, D., Castelletti, A., and Rizzoli, A. (2017). A hybrid signature-based iterative disaggregation algorithm for non-intrusive load monitoring. *Applied Energy*, 185:331–344.

Commons, W. (2017). File:openmp language extensions.svg — wikimedia commons, the free media repository. [Online; accessed 22-September-2017].

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A. (2011). *Combinatorial Optimization*, volume 21. Springer, 53113 Bonn, Germany, fifth edition.

Couprie, C., Farabet, C., Najman, L., and LeCun, Y. (2013). Indoor semantic segmentation using depth information. In *ICLR*.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2008). Image restoration by sparse 3d transform-domain collaborative filtering. In *Electronic Imaging 2008*, pages 681207–681207. International Society for Optics and Photonics.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2009). Bm3d image denoising with shape-adaptive principal component analysis. In *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *CVPR*.

Dalva, M. B., Weliky, M., and Katz, L. C. (1997). Relationships between local synaptic connections and orientation domains in primary visual cortex. *ScienceDirect*, 19(4):871–880.

Dauphin, Y., DeVries, H., Chung, J., and Bengio, Y. (2015). Rmsprop and equilibriated adaptive learning rates for non-convex optimization. *arXiv:1502.04390v1*, pages 1–10.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems*, pages 2933–2941.

Davis, T. A. (2006). Direct methods for sparse linear systems. *SIAM*.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of Association of Computing Machinery*, pages 1–9.

Domke, J. (2013). Learning graphical model parameters with approximate marginal inference. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 35(10):2454 – 2467.

Duarte Antunes, J. Hespanha, C. S. (2013). Stochastic hybrid systems with renewal transitions: Moment analysis with applications to networked control systems with delays. *SIAM J. Contr. Optimization*, 51(2):1481–1499.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Eisemann, E. and Durand, F. (2004). Flash photography enhancement via intrinsic relighting. In *ACM transactions on graphics (TOG)*, volume 23, pages 673–678. ACM.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745.

Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. *arXiv:1512.03965v4*, pages 1–33.

Erdogan, H. (2010). A tutorial on sequence labeling. International Conference on Machine Learning Applications.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, pages 625–660.

Erwig, M., Guting, R. H., Schneider, M., and Vazirgiannis, M. (1999). Spatiotemporal data types: An approach to modeling and querying moving objects in databases*. *GeoInformatica*, 3(3):1–22.

F. W. Nutter, J., Tylka, G. L., Guan, J., Moreira, A. J. D., Marett, C. C., Rosburg, T. R., Basart, J. P., and Chong, C. S. (2002). Use of remote sensing to detect soybean cyst nematode-induced plant stress. *Journal of Nematology*, 34(3):222–231.

Faghihi, J. and Ferris, J. M. (2000). An efficient new device to release eggs from heterodera glycines. *Journal of Nematology*, pages 411–413.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *exdb*, pages 1–15.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230.

Ford, S. D., Welsh, B. M., Roggemann, M. C., and Lee, D. J. (1997). Reconstruction of low-light images by use of the vector wiener filter. *JOSA A*, 14(10):2678–2691.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. (2008). Nonparametric bayesian learning of switching linear dynamical systems. *Neural Information Processing Systems*.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. (2009). Nonparametric bayesian learning of switching linear dynamical systems. *Advances in Neural Information Processing Systems*, pages 457–464.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. (2011). A sticky hdp-hmm with application to speaker diarization. *The Annals of Applied Statistics*, 5(2A):1020–1056.

Fox, J. M. and Whiteside, G. M. (1987). Warning signals for eruptive events in spreading fires. *PNAS*, 112(8):2378–2383.

Fung, J. and Mann, S. (2004). Using multiple graphics cards as a general purpose parallel computer: Applications to compute vision. *International Conference on Pattern Recognition, ICPR*, 1:805–808.

Garshasbi, S., Kurnitski, J., and Mohammadi, Y. (2016). A hybrid genetic algorithm and monte carlo simulation approach to predict hourly energy consumption and generation by a cluster of net zero energy buildings. *Applied Energy*, 179:626–637.

Gehring, J., Miao, Y., Metze, F., and Waibel, A. (2013). Extracting deep bottleneck features using stacked auto-encoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3377–3381. IEEE.

Gere, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *IEEE Journal on Neural Computation*, 12(10):2451–2471.

Ghahramani, Z. and Jordan, M. I. (1997). Factorial hidden markov models. Kluwer Academic Publishers, Boston, MA.

Glorot, X., Bardes, A., and Bengio, Y. (2011). Deep sparse rectifier networks. *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, JMLR W& CP*, 15:315–323.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9:249–256.

Goodale, M. A. and Milner, A. D. (1992). Separate visual pathways for perception and action. *Elsevier Science Publishers(UK).*

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Orzair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *arXiv:1406.2661v1*, pages 1–9.

Goodfellow, I., Vinyals, O., and Saxe, A. (2015). Qualitatively characterizing neural network optimization problems. *International Conference on learning and recognition*, pages 1–20.

Grabau, J. Z. (2011). Management strategies for control of soybean cyst nematode and their effect on the nematode community. Master's thesis, University of Minnesota, St Paul, MN 55108-6068.

Grant, M., Boyd, S., and Ye, Y. (2008). Cvx: Matlab software for disciplined convex programming.

Graves, A. (2014). Generating sequences with recurrent neural networks. *arXiv:1308.0850v5 [cs.NE]*, pages 1 – 43.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *IJCNN*, pages 1 – 8.

Guo, H., Qiu, C., and Vaswani, N. (2014). An online algorithm for separating sparse and low-dimensional signal sequences from their sum. *IEEE Trans. on Signal processing*, (62):4287 – 4297.

Gupta, S., Ray, A., Sarkar, S., and Yasar, M. (May 2008). Fault detection and isolation in aircraft gas turbine engines: Part i - underlying concept. *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, 222(3):307–318.

Gutmann, M. and Hyvarinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Journal of Machine Learning Research*, 9:297–304.

Hart, G. (1992). Nonintrusive appliance load monitoring. *Proceedings of IEEE*, 80(12).

Hendron, R. and Engebrecht, C. (2010). Building america house simulation protocols (revised). Technical report, National Renewable Energy Laboratory (NREL), Golden, CO.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580v1*, pages 1–18.

Huang, Y. and Yang, V. (2009). Dynamics and stability of lean-premixed swirl-stabilized combustion. *Progress in Energy and Combustion Science*, pages 293–364.

Hussain, A. K. M. F. (1983). Coherent structures - reality and myth. *Physics of Fluids*, 26(10):2816–2850.

Ioffe, S. and Segzedy, C. (2015). Batch normalization: Accelerating deep network training by reducing the internal covariate shift. *arXiv:1502.03167v3*, pages 1–11.

Jafri, R. and Arabnia, H. R. (2009). A survey of face recognition techniques. *Journal of Information Processing Systems*, 6(2):42–68.

Jain, R. K., Smith, K. M., Culligan, P. J., and Taylor, J. E. (2014). Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial monitoring granularity on performance accuracy. *Applied Energy*, 123:168–178.

Jain, V. and Seung, S. (2008). Natural image denoising with convolutional networks. *Neural information Processing Standard*, pages 1–8.

Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *IEEE*, pages 2146–2153.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Jiang, Z., Balu, A., Hegde, C., and Sarkar, S. (2017a). Collaborative deep learning in fixed topology networks. *arXiv:1406.6651v1[cs.LG]*, pages 1–21.

Jiang, Z., Liu, C., Akintayo, A., Henze, G., and Sarkar, S. (2017b). Energy prediction using spatiotemporal pattern networks. *Applied Energy*, 206:1022–1039.

Jiang, Z. and Sarkar, S. (2015). Understanding wind turbine interactions using spatiotemporal pattern network. In *ASME 2015 Dynamic Systems and Control Conference*, page V001T05A001. American Society of Mechanical Engineers.

Jin, X., Guo, Y., Sarkar, S., Ray, A., and Edwards, R. M. (2011). Anomaly detection in nuclear power plants via symbolic dynamic filtering. *IEEE Transactions on Nuclear Science*, 58(1):277–288.

Jin, X., Sarkar, S., Mukherjee, K., and Ray, A. (2009). Suboptimal partitioning of time-series data for anomaly detection. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 1020–1025. IEEE.

Jones, S. (2015). Convolutional autoencoders in python-theano-lasagne. https://swarbrickjones.wordpress.com/.

Jung, J. and Broadwater, R. P. (2014). Current status and future advances for wind speed and power forecasting. *Renewable and Sustainable Energy Reviews*, 31:762–777.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. *IEEE(CVPR)*, 756:470–857.

Kennel, M. B. and Buhl, M. (2003). Estimating good discrete partitions from observed data: Symbolic false nearest neighbors. *Phys. Rev. Lett.*, 91:084102.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *International Conference on Learning and Recognition*, pages 1–15.

Kleiminger, W., Beckel, C., and Santini, S. (2015). Household occupancy monitoring using electricity meters. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 975–986. ACM.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kolter, J. Z. and Jaakola, T. (2012). Approximate inference in additive factorial hidden markov models with application in energy disaggregation. *In Proceedings of the International Conference on Artificial Intelligence and Statistics, La Palma, Canary Islands*, pages 1472 – 1482.

Kolter, J. Z. and Johnson, M. J. (2011). Redd: A public data set for energy disaggregation research. *SustKDD workshop*, (1):1–6.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems, Lake Tahoe, Nevada*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. *NIPS*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012c). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, page 9.

Kulesza, T., Amershi, S., Caruana, R., Fisher, D., and Charles, D. (2014). Structured labeling to facilitate concept evolution in machine learning. *ACM*, pages 1 – 10.

Kullback, S. and Liebler, R. (1951). On information and sufficieny. *The annals of mathematical statistics*, pages 79–86.

Kwon, S.-D. (2010). Uncertainty analysis of wind energy potential assessment. *Applied Energy*, 87(3):856–865.

Laboratory, N. R. E. (2017). Nwtc information portal. https://nwtc.nrel.gov/Data.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning*, pages 282–289.

Lai, K., Bo, L., and Fox, D. (2014). Unsupervised feature learning for 3d scene labeling. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. *Journal of Machine Learning Research*, 15:29–37.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech and time-series. In *The Handbook of Brain Theory and Neural Networks*. MIT Press.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. *exdb*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proc. of IEEE*, pages 1–46.

LeCun, Y., Bottou, L., Orr, G. B., and Muller, K.-R. (1998b). Efficient backprop. Neural Networks: tricks of the trade.

Lee, C. and Landgrebe, D. (1993). Feature extraction and classification algorithms for high dimensional data. published, School of Electrical Engineering, Purdue University.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. *Neural information Processing Systems*, pages 1–8.

Lee, S.-W., Maik, V., Jang, J., Shin, J., and Paik, J. (2005). Noise-adaptive spatio-temporal filter for real-time noise removal in low light level images. *Consumer Electronics, IEEE Transactions on*, 51(2):648–653.

Li, B. (2017). 3d fully convolutional network for vehicle detection in point cloud. *IROS*.

Li, B., Zhang, T., and Xia, T. (2016). 3d vehicle detection from 3d lidar using fully convolutional network. *Robotics Science and Systems*.

Li, F. (2011). *A hybrid camera system for low-light imaging*. University of Delaware.

Li, H., Zhou, X., Jeffries, J. B., and Hanson, R. K. (2007). Sensing and control of combustion instabilities in swirl-stabilized combustors using diode-laser absorption. *AIAA*, 45(2):1–9.

Li, L., Zhang, Q., and Huang, D. (2014). A review of imaging techniques for plant phenotyping. *Sensors*, 14:20078–20111.

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv:1506.00019v4*, pages 1–38.

Liu, C., Ghosal, S., Jiang, Z., and Sarkar, S. (2016). An unsupervised spatiotemporal graphical modeling approach to anomaly detection in cps. *Proceedings of the International Conference on Cyber-physical Systems (ICCPS)*, pages 1 – 10.

Liu, G., Ouyang, M., Lu, L., Li, J., and Hua, J. (2015). A highly accurate predictive-adaptive method for lithium-ion battery remaining discharge energy prediction in electric vehicle applications. *Applied Energy*, 149:297–314.

Liu, H., Shi, J., and Erdem, E. (2010). Prediction of wind speed time series using modified taylor kriging method. *Energy*, 35(12):4870–4879.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *CVPR*, pages 3431–3440.

Lore, K. G., Akintayo, A., and Sarkar, S. (2017). Llnet: A deep autoencoder approach to natural low-light image enhancement. *Elsevier Journal of Pattern Recognition*, 61:650–662.

Lowe, D. G. (2004). Distinctive image features from scale-invariat keypoints. *IJCV*.

Loza, A., Bull, D., Hill, P., and Achim, A. (2013). Automatic contrast enhancement of low-light images based on local statistics of wavelet coefficients. *Elsevier Digital Signal Processing*, 23(6):1856–1866.

Ma, H., Mao, F., and Taylor, G. W. (2016). Theano-mpi: a theano-based distributed training framework.

Madsen, R. E., Kauchak, D., and Elka, C. (2005). Modeling word burstiness using the dirichlet distribution. *International Conference on Machine Learning, Germany*, pages 545–552.

Makonin, S., Ellert, B., Bajic, I. V., and Popowich, F. (2016). Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014. *Scientific Data*, 3(160037):1–12.

Makonin, S., Popowich, F., Bartram, L., Gill, B., and Bajic, I. V. (2013). Ampds: A public dataset for load disaggregation and eco-feedback research. In *Electrical Power & Energy Conference (EPEC), 2013 IEEE*, pages 1–6. IEEE.

Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. *ICCV*.

Martens, J. (2010). Deep learning via hessian-free optimization. *International Conference on Machine Learning*, pages 1–8.

Matsui, S., Okabe, T., Shimano, M., and Sato, Y. (2010). Image enhancement of low-light scenes with near-infrared flash images. In *Computer Vision–ACCV 2009*, pages 213–223. Springer.

Mckee, C. (2017). Cuda cores in video cards. https://www.lifewire.com/what-is-nvidia-cuda-834095.

Meyer, A. (2011-2012). Hmm and part of speech tagging. Lecture Note.

Minka, T. (2017). Lightspeed toolbox. https://github.com/tminka/lightspeed/. [Online; accessed 15-May-2017].

Mitchell, T. M. (2006). The discipline of machine learning. online.

Mitka, A. M. and Bart, R. S. (2015). Image-based phenotyping of plant disease symptoms. *frontiers in Plant Science*, 5(734):1–8.

Mivule, K., Harvey, B., Cobb, C., and Sayed, H. E. (2014). A review of cuda, mapreduce, and pthreads parallel computing models. *International Journal of Innovative Science, Engineering and Technology*, 1(8).

Mozer, M. C. (1989). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems Publications, Inc*, 3:340–381.

Mukherjee, K. and Ray, A. (2014). State splitting and merging in probabilistic finite state automata for signal representation and analysis. *Signal processing*, 104:105 –119.

Murphy, K. (2013). Hidden markov model (hmm) toolbox for matlab, 2005. *URL http://www. cs. ubc. ca/murphyk/Software/HMM/hmm. html*.

Nair, V., Thampi, G., and Sujith, R. (2014). Intermittency route to thermoacoustic instability in turbulent combustor. *Journal of Fluid Mechanics*, 756:470–857.

Nelson, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, pages 1–6.

Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transaction on Image Processing*, 14(9):1360 – 1371.

Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493.

Nvidia (2017). Cuda cores in video cards. https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-black/specifications.

Okuno, Y., Small, M., and Gotoda, H. (2015). Dynamics of self-excited thermoacoustic instability in a combustion system: Pseudo-periodic and high-dimensional nature. *Chaos*, 25(043107):1–6.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *Journal of Machine Learning Research*, 28:1–9.

Pathria, R. (1996). *Statistical Mechanics*. Butterworth-Heinemann, Oxford, UK, 2nd edition.

Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., and Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM transactions on graphics (TOG)*, 23(3):664–672.

Picheny, M. (2016). Look whos talking: Ibm debuts watson speech to text speaker diarization beta. https://www.ibm.com/blogs/watson/2016/12/look-whos-talking-ibm-debuts-watson-speech-text-speaker-diarization-beta/.

Pinheiro, P. and Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. *ICML*, pages 1 – 9.

Pisano, E., Zong, S., Hemminger, B., DeLuce, M., Johnston, E., Muller, K., Braeuning, P., and Pizer, S. (1998). Contrast limited adaptive histogram equalization image processing to improve the detection of simulated spiculations in dense mammograms. *Journal of Digital Imaging*, 11(4):193–200.

Pitsikalis, V., Kokkinos, I., and Moragus, P. (2003). Nonlinear analysis of speech signals: Generalized dimensions and lyapunov exponent. In *Proc. European Conf. on Speech Communication and Technology - Eurospeech 2003*, pages 1 – 4, Geneva, Switzerland.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., and Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368.

Poinsot, T. J., Trouve, A. C., Veynante, D. P., Candel, S. M., and Esposito, E. J. (1987). Vortex-driven acousticaly coupled combustion instabilities. *J. Fluid Mech.*, 177:265–292.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech proccessing. *Proceedings of the IEEE*, 77(2):257–286.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation earning with deep convolutional generative adversarial networks. *International Conference on Learning and Recognition*, pages 1–16.

Ralanamahatana, C. A., Lin, J., Gunopulos, D., and Keogh, E. (2005). *Mining time Series data*, volume 6925, chapter 1, pages 18–36. Springer Link.

Ranzato, M. A., Poultney, C., Chopra, S., and LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems*, 19:1137–1144.

Rao, C., Mukherjee, K., Sarkar, S., and Ray, A. (2009a). Statistical estimation of multiple parameters via symbolic dynamic filtering. *Signal Processing*, 89:981 – 988.

Rao, C., Ray, A., Sarkar, S., and Yasar, M. (2009b). Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns. *Signal, Image, and Video Processing*, 3:101–114.

Ray, A. (2004). Symbolic dynamic analysis of complex systems for anomaly detection. *SignalProcessing*, 84(7):1115–1130.

Redmon, J. C. (2017). Darknet. GitHub: https://pjreddie.com/darknet/hardware-guide/.

Richecoeu, F., L.Hakim, Renand, A., and Zimmer, L. (2012). Dmd algorithms for experimental data processing in combustion. pages 459–468. Center for turbulence Research, Proceedings of Summer Program.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. *International Conference on Machine Learning*, pages 1–8.

Rojas, R. (1996). *Neural Networks*. Springer-Verlag Berlin.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Roux, N. L. and Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20.6:1631–1649.

Ruder, S. (2016). An overview of gradient descent optimization algorithms*. *arXiv:1609.04747v1*, pages 1–12.

Rumelhart, D. E. and McClelland, J. L. (1986). *A framework for parallel distributed processing: Explorations in the microstructure cognition*, volume 1. Cambridge, MA.

Salakhutdinov, R. and Hinton, G. (2012). An efficient learning procedure for deep boltzmann machines. *Neural Computation*, 24(8):1967–2006.

Salakhutdinov, R. R. and Hinton, G. E. (2009). Deep boltzmann machines. *In Procs of the International Conference on Artificial Intelligence and Statistics*, 12:1–8.

Samad, T. and Annaswamy, A. M. (2011). The impact of control technology. *IEEE*.

Santoso, A., Nugroho, E., Suparta, B., and Hidayat, R. (2011). Compression ratio and peak signal to noise ratio in grayscale image compression using wavelet. *International Journal of Computer Science and Technology*, 2(2):1–10.

Sarkar, S. (2015). *Hierarchical Symbolic Perception in Dynamic Data Driven Application Systems*. PhD thesis, The Pennsylvania State University.

Sarkar, S., Damarla, T., and Ray, A. (2015a). Real-time activity recognition from seismic signature via multi-scale symbolic time series analysis(mstsa). In *American Control Conference,2015*, Chicago, Il.

Sarkar, S., Jiang, Z., Akintayo, A., Krishnamurthy, S., and Tewari, A. (2016). Probabilistic graphical modeling of distributed cyber-physical systems. In Song, H., Rawat, D., Jeschke, S., and Brecher, C., editors, *Cyber-Physical Systems: Foundations, Principles and Applications*, number 9780128038017, chapter 18, pages 265–286. Todd Green, 1st edition.

Sarkar, S., Lore, K. G., and Sarkar, S. (2015b). Early detection of combustion instability by neural-symbolic analysis on hi-speed video. *Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CoCo @ NIPS 2015)*.

Sarkar, S., Lore, K. G., Sarkar, S., Ramaman, V., Chakravarthy, S. R., Phoha, S., and Ray, A. (2015c). Early detection of combustion instability from hi-speed flame images via deep learning and symbolic time series analysis. *Annual Conference of the Prognostics and Health Management Management Society*, pages 1–10.

Sarkar, S., Mukherjee, K., Jin, X., and Ray, A. (March 2012). Optimization of symbolic feature extraction from time-series for classification. *Signal Processing*, 92(3):625–635.

Sarkar, S., Mukherjee, K., Sarkar, S., and Ray, A. (2000). Symbolic dynamic analysis of transient time series for fault detection in gas turbine engines. Technical Brief DS-11-1309, The Pennsylvania State University, University Park, PA 16802. To appear in J. Dyn. Sys. Meas. Control.

Sarkar, S., Sarkar, S., Mukherjee, K., Ray, A., and Srivastav, A. (December 2013a). Multi-sensor data interpretation and semantic fusion for fault detection in aircraft gas turbine engines. *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, 227(12):1988–2001.

Sarkar, S., Sarkar, S., Virani, N., Ray, A., and Yasar, M. (2014). Sensor fusion for fault detection & classification in distributed physical processes. *frontiers in Robotics and AI - Sensor Fusion and Machine Perception*.

Sarkar, S. and Srivastav, A. (2016). A composite discretization scheme for symbolic identification of complex systems. *Signal Processing*, 125:156 – 170.

Sarkar, S., Srivastav, A., and Shashanka, M. (2013b). Maximally bijective discretization for data-driven model of complex systems. *American Control Conference, Washington D. C.*, (978-1-4799-0178-4):2674–2679.

Sarkar, S., Venugopalan, V., Reddy, K., Ryde, J., Giering, M., and Jaitly, N. (2015d). Occlusion edge detection in rgbd frames using deep convolutional neural networks. *Proceedings of IEEE High Performance Extreme Computing Conference, (Waltham, MA)*.

Sarkar, S., Virani, N., Yasar, M., Ray, A., and Sarkar, S. (2013c). Proceedings of american control conference, washington, dc. In *Spatiotemporal Information Fusion for Fault detection in Shipboard Auxilliary Systems*.

Sarkar, S., Yasar, M., Gupta, S., Ray, A., and Mukherjee, K. (May 2008). Fault detection and isolation in aircraft gas turbine engines: Part ii - validation on a simulation test bed. *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, 222(3):319–330.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120v3*, pages 1–22.

Schaul, T., Antonoglou, I., and Silver, D. (2014). Unit tests for stochastic optimization. *arXiv:1312.6055v3*, pages 1–13.

Scherer, D., Muller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *Intenational Conference on Artificial Neural Networks, ICANN*, pages 1–10.

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28.

Schuler, C., Hirsh, M., Harmeling, S., and Scholkpf, H. (2014). Learning to deblur. *arXiv:1406.7444v1 [cs.CV]*, pages 1–28.

Segzedy, C., Liu, W., Jia, Y., Sermanet, P., and Reed, S. (2015). Going deeper with convolutions. *Computer Vision Foundation (CVPR)*, pages 1–9.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650.

Shao, H., Marwah, M., and Ramakrishnan, N. (2012). A temporal motif mining approach to unsupervised energy disaggregation applications to residential and commercial buildings. *1st International Workshop on Non-Intrusive Load Monitoring*, pages 1–7.

Siegelmann, H. T. and Sontag, E. D. (1991). Turing compatibility with neural nets. *Applied Mathematics Letters*, 4(6):77–80.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *NIPS*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutionaal networks for large scale image recognition. *International Conference on Learning and Recognition*, (arXiv:1409.1556v6):14.

Singh, A., Ganapathysubramanian, B., Singh, A. K., and Sarkar, S. (2015). Machine learning for high-throughput stress phenotyping in plants. *Trends in Plant Science, Elsevier*, 21(2):110–124.

Sipser, M. (2013). *Introduction to the Theory of COMPUTATION*. Number 13. Cengage Learning, 3rd edition.

S.Ji (2010). 3d convolutional neural networks for human action recognition. *ICML*.

Smola, A. and Vishawanathan, S. (2008). *Introduction to Machine Learning*. Cambridge University Press, New York.

Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. *Annual Conference on Learning Theory*, pages 545–560.

Srivasta, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Srivasta, N. and Salakhutdinov, R. (2012). Multimodel learning with deep boltzmann machines. *Neural Information Processing Systems*, pages 1–9.

Subbu, A. and Ray, A. (2008). Space partitioning via hilbert transform for symbolic time series analysis. *Applied Physics Letters*, 92:084107.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning, JMLR*, 28.

Syed, F. A. (2015). Development of an automated system for extraction and quantification of soybean cyst nematode (scn) eggs and cysts. Master's thesis, University of Illinois at Urbana-Champaign, Technical Systems Management Department.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *CVPR*, page 9.

Tascikaraoglu, A., Sanandaji, B. M., Poolla, K., and Varaiya, P. (2016). Exploiting sparsity of interconnections in spatio-temporal wind speed forecasting using wavelet transform. *Applied Energy*, 165:735–747.

Team, T. T. D., Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brbisson, A., Breuleux, O., Carrier, P.-L., Cho, K., Chorowski, J., Christiano, P., Cooijmans, T., Ct, M.-A., Ct, M., Courville, A., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S.,

Dinh, L., Ducoffe, M., Dumoulin, V., Kahou, S. E., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I., Graham, M., Gulcehre, C., Hamel, P., Harlouchet, I., Heng, J.-P., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrancois, S., Lemieux, S., Lonard, N., Lin, Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P.-A., Mastropietro, O., McGibbon, R. T., Memisevic, R., van Merrinboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., tienne Simon, Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A python framework for fast computation of mathematical expressions.

Teh, Y. W. (2007). A tutorial on dirichlet processes and hierarchical dirichlet processes.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *SCIENCE*, 331(1192788):1279–1285.

Thoma, M. (2016). Soybean nematode management: Field guide. online.

Trahanias, P. and Venetsanopoulos, A. (1992). Color image enhancement through 3-d histogram equalization. In *Pattern Recognition, 1992. Vol. III. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on*, pages 545–548. IEEE.

Tranter, S. and Reynolds, D. (2006). An overview of automatic speaker diarisation systems. *IEEE Transactions on Speech, Audio and Language Processing: Special Issue on Rich Transcription*, pages 1557–1565.

Tsatsos, M. (2006). *Theoretical and Numerical Study of the Van der Pol equation.* PhD thesis, Aristotle University of Thessaloniki, School of Sciences.

Tylka, G. L. (2008). Soybean nematode management: Field guide. online.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International conference on Machine Learning*, pages 1096–1103.

Volk, M. L., Tocco, R. S., Bazick, J., Rakoski, M. O., and Lok, A. S. (2012). Hospital readmissions among patients with decompensated cirrhosis. *Am. J Gastroenterol*, 107(2):247–252.

Wang, G. and Giannakis, G. B. (2016). Solving random systems of quadratic equations via truncated generalized gradient flow. *arXiv:1605.08285v1*, pages 1–30.

Wang, H., Klaser, A., Schmid, C., and Liu, C. L. (2011). Action recognition by dense trajectory tracking. *Proc. of the CVPR*, pages 3169–3176.

Wang, J.-Z., Wang, Y., and Jiang, P. (2015). The study and application of a novel hybrid forecasting model–a case study of wind speed forecasting in china. *Applied Energy*, 143:472–488.

Wang, Y.-H. (2010). Tutorial: Image segmentation. Online.

Wang, Z., Bovik, A., Sheik, H., and Simoncelli, E. (2004). Image quality assessment: From error visibility to structural similarity. *IEE Trans. Image Process.*, 13(4):600–612.

Wilks, S. (1963). *Mathematical Statistics*. John Wiley, New York, NY, USA.

Wolpert, D. H. and Macready, W. G. (2012). No free lunch theorems for optimization. *http* : $//georgemaciunas.com/wp - content/uploads/2012/07/Wolpert\_NLFoptimization - 1.pdf$.

Wu, Z., Wang, X., Jiang, Y.-G., Ye, H., and Xue, X. (2015). Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. *ACM*.

Xie, J., Xu, L., and Chen, E. (2012). Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 341–349.

Zeifman, M. and Roth, K. (2011). Non-intrusive appliance load monitoring (nialm): Review and outlook*. *International Conference on Cosumer Electronics*, pages 1–27.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv:1212.5701v1*, pages 1–6.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. *European Conference on Computer Vision, Zurich Switzerland*, (8689):818–833.

Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2016). Loss functions for neural neiworks for image processing. *arXiv:1511.08861v2*, pages 1–16.

Ziel, F., Croonenbroeck, C., and Ambach, D. (2016). Forecasting wind power–modeling periodic and non-linear effects under conditional heteroscedasticity. *Applied Energy*, 177:285–297.

Zuluaga, C. D., Álvarez, M. A., and Giraldo, E. (2015). Short-term wind speed prediction based on robust kalman filtering: An experimental comparison. *Applied Energy*, 156:321–330.