

Feature extraction by deep learning for emotion estimation



Motaz sabri

Department of Engineering
Hiroshima University

This dissertation is submitted for the degree of
Doctor of Philosophy

Computer engineering

August 2018

After many sincere prayers and hard work, we reached this far together, you were here, are here and will be here in my heart and mind. To my loving parents and brother ...

Declaration

I declare that this thesis is my original work, except where stated. The thesis has been composed by myself and represents my own work. Any data, idea or opinion taken from an external source is appropriately acknowledged in the text. The work presented has not been submitted for any other degree or professional qualification. Technical and editorial guidance were provided by Professor Takio Kurita. I designed all the experiments presented in this thesis with advice from colleagues and I analysed all the experimental data. Some of the material included in the thesis was presented at the following journals and conference:

- Journal: Elsevier Neurocomputing; Accepted; 2018.
- Journal: Kansei Engineering International Journal; Japan; 2018.
- Journal: Transactions on Information and Systems, Institute of Electronics, Information and Communication Engineers; Japan; 2017.
- Conference: The International Workshop on Frontiers of Computer Vision; South Korea; 1-3 Feb 2017.

Motaz sabri
August 2018

Acknowledgements

I am grateful to ministry of education, culture, sports, science, and technology (MEXT) for the opportunity to do research in Japan, for the funding.

To **Professor. Takio Kurita** I am grateful for his support during my graduate study years and the opportunity to be in his lab in the early years and guiding me through the ups and downs of experiments designs and phenomena understanding and simplification.

To **Fellow research, master and PhD students** within our group who kept each other motivated when we felt stranded.

I also must thank all my **Hiroshima university educators**, all the dedicated and talents faculty who invested in us and encouraged us to think outside the box and to take risks.

To **my family, friends and kindred spirits** – there is no need for words. you are always with me.

They made all the difference in the last couple of years – without their moral support the thesis would not have been completed.

Abstract

Optimizing the extraction of feature sets for classification tasks is still a fundamental and challenging problem in the area of machine learning. For Mammals to perform classification tasks we use hierarchical features and efficient coding in our visual pathway. Information in visual system is encoded using distributed coding schemes and later the sparse coding is utilized. We propose multiple architectures to extract features for emotional analysis that encode the information according to a specific activation profile. We show how our models much like the visual system, can learn distributed coding in lower layers and sparse coding in higher layers for emotional analysis. Feature extraction for emotion related tasks must ensure high accuracy in locality and activation. Our models can dynamically extract features and perform classification and ranking for many commonly used datasets. As for emotional classification through facial expressions, we introduce regularization through noisy training that helped internal representation of learned features to emerge and increased sparsity of hidden units. This regularization also improved the network generalization through automatic structuration. For our ranking models that estimate emotion intensity, they extracted the sequential relationship in the temporal domain that appears due to the natural change of facial expression during an emotional transition. The output ranking score can be directly used for intensity estimation for all emotional states. This allows an early detection of an emotional transition and estimation of intensity of an emotion considering temporal information. Beside facial expressions for emotion analysis, we proposed an eye tracking that can be used to understand person emotions through pupil movement. Through Gabor filter, we detected the intensity change of the gaze edge by applying Gabor filter and its second moment matrix. Eigenvalues inequality of this matrix with a given local intensity-based threshold defines the prevailing directions of the neighboring gradients and the amount by which these directions are coherent at a certain point. This allows accurate estimation of the gaze direction.

We find that all CNN features can be used for knowledge representation purposes both by their activation and locality, doubling the information a single CNN feature may provide. We also study the levels of importance for these features, and propose approaches and structures to discard most of it. All these insights have a direct application to the generation of CNN embedding spaces.

Table of contents

List of figures	xv
List of tables	xvii
1 Introduction	1
1.1 Motivation	1
1.2 Structure of thesis	2
2 Generalization effect of Additive noise on MLP with AutoEncoder	5
2.1 Introduction	5
2.2 Regularized Backpropagation	6
2.3 Multi-layered noise enabled neural network	8
2.3.1 Network Modeling	8
2.3.2 Weight Update Rules	11
2.3.3 Additive noise -Hidden layer	13
2.3.4 Additive noise -Input layer	16
2.4 Experiment	19
2.5 Conclusion	26
3 Enhancing emotions estimation accuracy by additive noise	27
3.1 EmotionUnderstanding	27
3.2 Related work	29
3.2.1 Inception model	29
3.2.2 Regularization: Noise injection	30
3.2.3 Emotion recognition by using CNN	30
3.3 Proposed Method	31
3.3.1 Proposed network architecture	31
3.3.2 Training with noise	32
3.3.3 Visualizing features	33

3.4	Experiments	35
3.4.1	Dataset	35
3.4.2	Single emotion expert case	36
3.4.3	Multiple emotions expert case	39
3.5	Discussion	43
3.6	Conclusion	46
4	Emotions intensity estimation by Siamese and Triplet networks	47
4.1	IntensityEstimation	47
4.2	Related work	49
4.2.1	Expressions Intensity Estimation	49
4.2.2	Siamese Network	49
4.2.3	Triplet Network	50
4.2.4	Micro-Expressions	50
4.3	Proposed method	51
4.3.1	Data Setup	51
4.3.2	Siamese and triplet networks	52
4.3.3	Visualizing features	53
4.4	Experiments	55
4.4.1	Dataset	55
4.4.2	Evaluation measures	57
4.4.3	Experiment 1- CK Dataset:	60
4.4.4	Experiment 2- MUG Dataset:	61
4.4.5	Experiment 3- MMI Dataset:	64
4.4.6	Experiment 4- Micro-Expressions:	64
4.5	Discussion	65
4.6	Conclusion	72
5	Dynamic eye tracking through Gabor filters	75
5.1	Related Work	77
5.2	Proposed Methodology	78
5.3	Experiments	81
5.3.1	Evaluation data	81
5.3.2	Evaluation Criteria	82
5.3.3	State of the art algorithms	82
5.4	Discussing Results	83
5.5	Conclusion	85

6 Conclusion	89
References	93
Appendix A	109
A.0.1 Additive Noise Update rule for u_{hi} - Equation 2.18	109
A.0.2 Expectation of Equation $\frac{\partial \tilde{\ell}_1}{\partial u_{hi}}$ - Equation 2.20	110
A.0.3 Expectation of Equation $\frac{\partial \tilde{\ell}_2}{\partial u_{hi}}$ - Equation 2.20	111

List of figures

2.1	AutoEncoder with MLP	9
2.2	AutoEncoder(s) with MLP	20
2.3	Accuracy and loss Charts under noise Injection location change: Single AE	22
2.4	Accuracy and loss Charts under noise Injection location change: Stacked AE	23
2.5	Feature Localztion	23
2.6	Histogram Charts	24
2.7	TSNE with noise placement	25
3.1	Stacked Inception model	31
3.2	Sample of Emotional Class activation mapping	34
3.3	Emotional Feature mapping	34
3.4	KDEF Dataset Samples	36
3.5	CK+ Dataset Samples	37
3.6	Experimental Feature Maps for KDEF and CK+	40
3.8	Training and Testing loss for KDEF with noise injection	40
3.7	Experimental Class activation Maps for KDEF and CK+	41
3.9	Accuracy for KDEF with noise injection	43
3.10	Training and testing loss for CK+	44
3.11	Training and testing accuracy for CK+	44
4.1	The proposed architecture shows the Siamese and triplet models connected to the exponential loss, the kernel size is shown on every block and S represents the Stride and P represents the Padding.	51
4.2	The organization of training data on the ranking of intensities of all emotional transitions for CK, MUG and MMI datasets. The sequence on top shows how Siamese pairs are created while the sequence at the bottom shows how triplet items are created. Every sequence starts from an emotional peak and ends at another emotional peak passing through neutral state.	52
4.3	Emotional class activation for intensity Estimation	54

4.4	Emotional feature mapping for intensity Estimation	55
4.5	Samples for CK Dataset	56
4.6	Samples for MUG Dataset	57
4.7	Samples for MMI Dataset	58
4.8	Samples for CASME Dataset	59
4.9	Ground Truth Illustration Diagram	59
4.10	Loss chart for CK Dataset	62
4.11	Loss Chart for MUG Dataset	63
4.12	Loss chart for MMI dataset	66
4.13	Feature Localization for CASME Dataset using Siamese Network	67
4.14	Feature Localization for CASME Dataset using Triplet Network	68
4.15	Intensity Emotional Class Activation Mapping for CK Dataset-Siamese	69
4.16	Intensity Emotional Class Activation Mapping for CK Dataset-Triplet	69
4.17	Intensity Emotional Feature Mapping	70
4.18	Intensity Emotional Class Activation Mapping for MUG Dataset-Siamese	71
4.19	Intensity Emotional Class Activation Mapping for MUG Dataset-Triplet	71
4.20	Intensity Emotional Class Activation Mapping for MMI Dataset-Siamese	73
4.21	Intensity Emotional Class Activation Mapping for MMI Dataset-Triplet	73
5.1	Work flow for Gaze tracking	76
5.2	Illustration for Parameters setting.	80
5.3	Samples from 1 Video 50 people Dataset- Brooklyn	82
5.4	Samples of eye tracking using Brooklyn branch of the Dataset	84
5.5	Samples of eye tracking using London branch of the Dataset	85
5.6	Gaze Detection comparisons for Brooklyn branch of Dataset	86
5.7	Gaze Detection comparisons for London branch of Dataset	87

List of tables

1	Accuracy under noise Injection location change: Single AE	21
2	Accuracy under noise Injection location change: Stacked AE	22
1	KDEF Noise placement results	38
2	CK+ Noise placement results	39
3	Confusion Matrix for KDEF without noise injection	39
4	Confusion Matrix for KDEF with Case 1 of noise injection	42
5	Confusion Matrix for KDEF with Case 2 of noise injection	42
6	Confusion Matrix for CK+ without noise injection	42
7	Confusion Matrix for CK+ with case 1 of noise injection	43
8	Confusion Matrix for CK+ with case 2 of noise injection	45
1	Results compariosons for CK dataset	61
2	Results comparisons for MUG Dataset	64
3	Results comparisons for MMI dataset	65
1	Accuracy and error comparisons for Brooklyn branch of Dataset	84
2	Accuracy and error comparisons for London branch of Dataset	85

Chapter 1

Introduction

1.1 Motivation

Smart technologies are shaping the way we interact with our surroundings. A successful smart technology is characterized by its ability to autonomously adapt to the steadily changing needs of interacting subject. Understanding the subject state, motivation, and emotion proposes an important contextual information for a system interacting with humans. Once a system performs an action it must know that action is perceived by the user and consequently it can adjust its behavior to the user's needs and fine-tune its actions.

Being of great importance to human-machine interactions, emotions are irreplaceable factor in our everyday routines. Many attempts to engage in emotion recognition to optimize affective knowledge integration into systems design. Thus, the term Affective Computing was proposed to describe the computation that relates to, arises from, or deliberately influences emotion or other affective phenomena [1]. The community of human computer interaction developed various concepts, models and frameworks for affective systems and devices. The research must grow to encompass the area of emotional awareness, where the goal of the system is no longer to produce a definition of an emotional state, but also to extract information about the semantics and pragmatics of the dialogue. It has become clear that automatic recognition of interactive affection in visual world is no longer a luxury that these systems can dispense with; it is a necessary ingredient for systems that adapt to the users, understand them and react appropriately.

One issue stands on the way of a robust emotional understanding is the generalization, since emotion research usually is done in strict conditions, the scientific body of knowledge on emotional reactions in real life situations is still very fragmentary. There is a lot to consider when designing a model that can be used for daily activities such as different ways to express an emotion and the amount of expression that the model can observe to detect

a transition into an emotional state. Many of the efforts in this field focused on pictures rather than videos neglecting very valuable temporal information. This is partially due to the absence of annotated training data and due to the difficulties, that are paired with interactive emotions understanding in the wild such as occlusion and spontaneous expressions.

Another issue is the discrete understanding of emotions. Many of the efforts in this field focused on pictures rather than videos neglecting very valuable temporal information. This is partially due to the absence of annotated training data and due to the difficulties, that are paired with interactive emotions understanding in the wild such as occlusion and spontaneous expressions. This makes it difficult to collect real-life data and to do research under real-life situations. It's know that such understanding in real life is complicated due to many contextual aspects to be considered, so many possible triggers for human state change, motivation and feelings, and so many interaction options to be considered, this limits the creation of an artificial emotional awareness without considering sensible assumptions about subject's mental state without limiting the setting to certain situations with well-known behavioral pattern and affective triggers.

The generalization and discrete interpretation problems has been discussed in the literature. Many approach such as machine learning have been utilized to understand emotions for years. Neural network for instance is one of the leading architectures in machine learning. Acting as general approximators with ability to observe numerous variables before making an action, Neural networks are valuable tool to analyze emotions once a reliable design is proposed. Thinking of the perfect design comes by understanding how a similar system works. When human attempts to understand the emotional state of a person next to him, a set of neurons trigger in our brain. The characteristics of those neurons vary depending on their neighboring neurons, but they work together into stimulating an emotional understanding. Consequently, this thesis aims to help overcoming understanding emotions through the extracted features through neural networks. The models we propose observe and analyze different aspects and perspectives allowing richer emotional understanding. We also regularize the training process by injecting noise which results in more salient features being more localized and accurately generated.

1.2 Structure of thesis

The well-known methods that have been researched to interpret human transitional behaviors observe certain physiological and visual parameters. Many disciplines must be taken into account considering aspects that are related to the states and triggers. Knowledge in related aspect is required for designing the preceptors as well as developing the appropriate algo-

rithms for feature extraction and classification. Monitoring the learning and understanding process and its representation offer valuable information in how an estimated is being evaluated. Consequently, aspects of emotions are considered when developing feature extractions and emotion classification algorithms. The pipeline developed in this work contains all steps necessary for providing an adaptive interpreter with the needed information to analyze emotions. The preceptor must capture valuable features that build up an emotion and avoid details that are nonconstructive for the process. Visualization of the learning process must be introduced to make sure knowledge is being accumulated and structures correctly. The human brain has been doing this task since we were born and building neural network models inspired by well-studied brain structures will produce valuable results.

Chapter 2 gives a short overview of the neural networks and the way knowledge is extracted and used for tasks such as classification. Regularization effect in the learning process is also discussed in this chapter through analyzing the effect of noises added to hidden units of deep network models. it concludes that internal representation of learned features emerges and sparsity of hidden units increases when independent Gaussian noises are added to inputs of hidden units during the deep network training. This is expected to improve the generalization ability of the network through this automatic structuration by adding the noises. The findings are confirmed through experiments on a classification task and through mathematical modeling of the learning process with more details included in Appendix 1. Chapter 3 follows with more emotional oriented neural network model that investigate feature localization abilities upon injecting noise into the convolutional neural network. In this chapter we propose a model that classifies the 7 human emotional states based on facial expressions and it is shown to perform better than the earlier convolutional neural networks. After this follows analysis of the internal representation of learned features that emerge beside more accurate localization of those features that appear when independent Gaussian noises are added to certain joints during the deep network training. The focus here is the observation that the weights after the noise contaminated units lead to output that is more definite. Such behavior improves the network generalization through automatic structuration. Implications for designing affective systems are drawn and requirements for affective interpretations are highlighted, which is used in next chapter.

Chapter 4 extends the extraction of local features to sequential relationship in the temporal domain that appears due to the natural onset apex offset during an emotional transition. Besides modeling analysis, we discuss the importance of how data feeding is important to the learning and understanding of emotions. We discuss the proposed model's abilities to learn internal representation of extracted features. More accurate localizations of those features appear with training.

Chapter 5 discusses another aspect of emotion understanding which is the eye movement. the problem of gaze-tracking and blink-detection under illumination variation and unrestricted subject movement is addressed. A method of gaze direction tracking in real time video stream is proposed. Gabor filter and its function's second moment matrix is used in the proposed model. It is shown that the Eigenvalues of this matrix and its inequality with a given local intensity-based threshold defines the prevailing directions of the neighboring gradients and the amount by which these directions are coherent at a certain point. The Automatic setting of the threshold is discussed. The evaluations of the developed systems have been performed by variety of datasets.

Chapter 6 provides proof of the applicability of the developed approaches to understanding and estimating emotions and representing affective states. The potentials of applications and benefits of such models are discussed in this chapter

Appendix A gives full mathematical definition of the standard and regularized learning of the proposed deep models in chapter 2. The update rules with their noise-enabled counter parts are proposed in this appendix highlighting the benefits of noisy training for the generalization and sparsity of the proposed models.

Chapter 2

Generalization effect of Additive noise on MLP with AutoEncoder

2.1 Introduction

Modern computing systems are becoming exceptionally complex to keep up with increasing performance demands of applications. The depth of those systems is increasing to match the new demands. This unprecedented scaling requires more reliable techniques to enhance learning those systems. Since the neural network regularization has been introduced, many techniques have been applied to generalize networks beyond the training data including L1 and L2 regularization, dropout [41] and preventing the co-adaptation of detectors feature [42, 4]. Many regularizing methods focus on optimizing network target function during the training [7, 8, 41, 6, 5]. One commonly used approach is introducing training criterion that discourages converging the network into building a complicated solution and suppresses the number of effective weights by directing the learning algorithm towards a solution with the least possible amount of zero weights such as weight decay [36]. Other approaches encourage less co-adapting between units and reduces overfitting such as Dropout [4].

One more powerful method of improving neural network generalization ability is noise injection. Kurita et al. [44] inspected the backpropagation learning behavior while injecting noises into the inputs of the hidden units during MLP training. They concluded that the connection from hidden units to output units holds smaller weights while the connections from input units to hidden units holds larger weights values. They confirmed the network ability of automatically structure itself by simply adding the noises. They conducted pattern classification and logic Boolean function learning to confirm their results. Kartik et al. [45] introduced the Noisy convolutional neural network algorithm which utilizes noisy expectation maximization result to generate a hyperplane in noise space that extracts helpful noise. They showed that injecting noise into neural network boosts the backpropagation training of a neural network.

Poole et al. [12] injected multiple noise configuration into neural networks and studied the learning process. They also analyzed generalization ability after injecting noise into networks with de-noising, contractive, and sparse AutoEncoders. They showed that noisy learning outperforms de-noising learning. They tested their noise scheme against MNIST, and CIFAR-10 and achieved comparable results. However, they haven't concluded the place in which the noise is best to be injected and the reason why the noise is proven to enhance the training process.

Motivated by the need of expanding neural network generalization ability, we analyze a simple technique of selectively injecting Gaussian noise to multiple joints within a deep neural network. We inspect the relation between the network learning and noise injection mathematically. We also conduct pattern classification experiments that visualize the internal representation of the network and its hidden unit activation sparsity. Our mathematical modeling clarifies why learning process improves upon noise injection into specific joints within the network. Our experiments steadily result in lower loss and better accuracy upon injecting Gaussian noise into selected joints of neural network to classify MNIST [13] digits. Our experiments covered single layered AutoEncoder and Stacked AutoEncoder both connected to an MLP during separate experiments of classifying MNIST digits which come with different handwriting styles. This matches the finding of many researches that inject noise to simple neural networks.

The contribution of this research is scoped towards showing improvement to the classification accuracy of deep network structures with standard gradient descent upon selective noise injection-point. To the best of our knowledge, our noise injection placement and tuning have not been used before in the training of deep neural networks. It shows better visualization for the hidden layer representation and more sparsity for the hidden unit's activation when compared to noiseless case. We hope other researchers will utilize this method to achieve similar improvements in other research schemes.

2.2 Regularized Backpropagation

We analyze the average behavior of the backpropagation learning to noise injected into specific joints of AutoEncoders connected to multilayer perceptron (MLP). Optimizing MLP for learning is based on selecting the suitable architecture and the connection weights via the minimizing training error and a penalty term [14]. Many techniques have been proposed to extend MLP abilities by additional components. Kurita et al. introduced AutoEncoder with competitive hidden units with classifiers as its input [15]. This formed a mixture of classifier with ability of self-organizing in which one of the classifiers is selectively activated

according to the input feature. Vincent et al. highlighted a method for building deep networks based on stacking layers of de-noising AutoEncoders [16]. The AutoEncoders reduce the dimensionality of input data to a smaller-dimensional code space at the hidden layer. Such reduction drops features with less importance and highlights the feature with higher value. This increases the network ability to observe features that MLP alone wasn't able to learn and ultimately improves learning and generalization ability [17, 18].

Interfering with the training process of a deep neural network has many shapes and has been shown to improve the performance of learning process [52]. Random Noise injection to the weights or the hidden units has been utilized in many neural network researches for many years [53, 54, 38, 16]. Kurita et al. [44] injected noise into the hidden layers of an MLP and showed the network ability to get automatically structurized by simply adding the noises and therefore improving the generalization ability of the network. Adding more randomness gives higher potential for escaping local minima or passing through the early training phase in the most optimum phase.

Zeyer Et al.[54] introduced a low-overhead technique of adding gradient noise for neural network which is found to avoid overfitting and result in lower training loss. Their method allows a fully-connected, poorly initialized 20-layer deep network to be trained with standard gradient descent. They achieved improvements for many complex models. They were able to achieve a 72% reduction in error rate over a carefully-tuned baseline on a challenging question-answering task. In a parallel experiment, they were able to double the number of accurate binary multiplication models learned across 7,000 random restarts. The noise injection scheme recently started tackling modern deep networks and its advantages haven't been fully documented in terms of mathematical representation and experiments variety [5, 54].

Guided by this fact we inspect a network with AutoEncoders and MLP behavior of learning update rules and noise injection mathematically. We assume the AutoEncoders ability to reduce dimensionality and highlight useful features will be strengthened due to noise injection. We aim to clarify why injecting noise into certain joints of network with AutoEncoders and MLP benefits the learning process. The contaminated neural network training proposed in this chapter was based on well-known noise injection theory in the neural computing community [16, 23, 12]. The visualizations of network hidden layers weights and activation sparsity were obtained to show the feature representation considering the location of noise injection. Experiments on MNIST [13] highlight benefits of using noisy AutoEncoders by learning useful features for classification. The effect of adding random Gaussian noise are predicted theoretically and proven practically through network learning updated rules analysis and experiments. We assumed adding Gaussian noise effect will

be comparable with well-known advantages of stochastic gradient descent as a learning algorithm by reducing overfitting.

2.3 Multi-layered noise enabled neural network

In this chapter, we theoretically and practically evaluate the effect of injecting noise into hidden and input layers of single layered and stacked AutoEncoders each connected to MLP neural network. The input data passes through an AutoEncoder first. The AutoEncoder learns detailed and useful features of the input data, in our case, the MNIST database of handwritten digits classification[13]. The hidden layer of the AutoEncoder is connected to an MLP that decides which class each input data belongs to based on a teacher signal.

In order to enable such architecture to realize the variety of hand writing styles, the AutoEncoder must have the ability to select one of the classifier depending on a special feature within each input. The AutoEncoder is trained to reduce the dimension of the representation of the input data by considering the outputs of the hidden layer. Figure 1 shows the symbolic variables of the network. As referred by Kurita et al. this structure introduces competition among the units in the hidden layer [15] and their outputs can be regarded as the signal for selecting the classifiers. AutoEncoders main training criterion is minimizing the reconstruction error with respect to some loss. We will describe the loss functions we use in our experiment in the next section.

2.3.1 Network Modeling

Objects that lay within the same category can be classified based on special features within them. When objects have variation within the same feature representation then the training process becomes more complicated. Mutli-shaped representations can be encoded and decoded by AutoEncoders to highlight unclear features and be classified later by logistic units. Each classifier receives the same input vector from the AutoEncoder hidden layer and outputs the classification result as an output vector. The output of the total network is computed as the weighted sum of the outputs of the classifiers. This means that the last layer of the network works like a selector of classifiers. Our network classifies K classes defined in the set $C = \{1, 2, \dots, K\}$.

In our module definition $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_I]^T$ represents the input feature vector of the classifier and $\mathbf{q} = [q_1 \ q_2 \ \dots \ q_H]^T$ is the bias vector of the hidden layer and

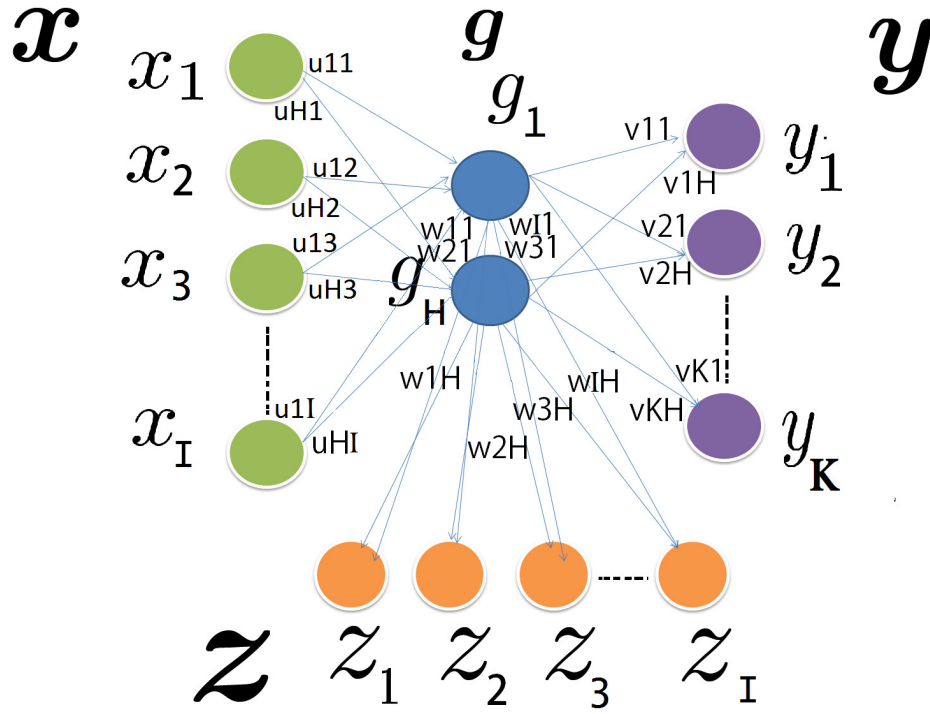


Fig. 2.1 AutoEncoders have proven its ability to highlight features that can't be identified by other neural network structure, Linking the classifier to AutoEncoder hidden layer allows the low dimension features to be extracted

$\mathbf{r} = [r_1 \ r_2 \ \dots \ r_I]^T$ be the bias vector of the output layer of the AutoEncoder and $\mathbf{s} = [s_1 \ s_2 \ \dots \ s_K]^T$ be the bias vector of the output layer of the network.

Let u_{hi} represents the weight of the connection between the i -th input unit and the h -th hidden unit therefore we define $\mathbf{u}_h = [u_{h1} \ u_{h2} \ \dots \ u_{hI}]^T$. Similarly let w_{ih} represents the weight of the connection between the h -th hidden unit and the i -th output unit of the AutoEncoder therefore we define $\mathbf{w}_i = [w_{i1} \ w_{i2} \ \dots \ w_{iH}]^T$. Let v_{ik} represent the weight of the connection between the h -th hidden unit and the k -th output unit of the network, therefore we define $\mathbf{v}_k = [v_{k1} \ v_{k2} \ \dots \ v_{kH}]^T$.

Lets define the weights vectors as matrices to define the activation functions, the matrix U is given by:

$$U = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_H]^T$$

the matrix W is given by:

$$W = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_I]^T$$

and the matrix V is given by:

$$V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_K]^T$$

To define the activation function of the AutoEncoder hidden layer let's define η_h as shown in equation 2.1:

$$\eta_h = \mathbf{u}_h^T \mathbf{x} + q_h \quad (2.1)$$

we define the hidden layer output vector \mathbf{g} as function of input \mathbf{x} as shown in equation 2.2:

$$\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}) \quad g_2(\mathbf{x}) \quad \dots \quad g_H(\mathbf{x})]^T = f(U\mathbf{x} + \mathbf{q}) \quad (2.2)$$

accordingly the hidden units have the sigmoid activation described by equation 2.3:

$$g_h(\mathbf{x}) = f(\eta_h) = \frac{1}{1 + e^{-\eta_h}} \quad (2.3)$$

Similarly to define the activation function of the output layer of the AutoEncoder let's define $\xi_i = \mathbf{w}_i^T \mathbf{g} + r_i$

Given the output vector of AutoEncoder \mathbf{z} as function of input \mathbf{g} as follows:

$$\mathbf{z}(\mathbf{g}) = [z_1(\mathbf{g}) \quad z_2(\mathbf{g}) \quad \dots \quad z_I(\mathbf{g})]^T = f(W\mathbf{g} + \mathbf{r})$$

accordingly the output units of the AutoEncoder have the linear activation of $z_i(\mathbf{g}) = f(\xi_i) = \xi_i$.

To define our activation function of the network output let's define $\rho_k = \mathbf{v}_k^T \mathbf{g} + s_k$

we define the output vector \mathbf{y} as function of AutoEncoder hidden layer output \mathbf{g} as follows:

$$\mathbf{y}(\mathbf{g}) = [y_1(\mathbf{g}) \quad y_2(\mathbf{g}) \quad \dots \quad y_K(\mathbf{g})]^T = f(V\mathbf{g} + \mathbf{s})$$

accordingly the output units of the network have the softmax activation as shown in 2.4:

$$y_k(\mathbf{g}) = f(\rho_k) = \frac{e^{\rho_k}}{\sum_{l=1}^K e^{\rho_l}} \quad (k = 1, 2, 3, \dots, K) \quad (2.4)$$

Let $\mathbf{t} = [t_1 \quad t_2 \quad \dots \quad t_K]^T \in \{0, 1\}^K$ denote a binary vector composed of teacher signals in which t_k equals to 1 if the current input vector is classified under the k -th class and 0 otherwise.

The training data $D = \{ \langle \mathbf{x}_p, t_p \in C \rangle \mid p = 1 \dots N \}$ where N is the number of samples. Each set of N input samples comes with its own output set and training signals, therefore the matrix X is given by:

$$X = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_N]^T$$

The n -th η term is given as η_{nh} and outputs of the hidden layer given input \mathbf{x}_n is given by $\mathbf{g}_n = \mathbf{g}(\mathbf{x}_n)$, therefore the Matrix G is given by:

$$G = [\mathbf{g}_1 \quad \mathbf{g}_2 \quad \dots \quad \mathbf{g}_N]^T$$

outputs of the AutoEncoder given output of hidden layer \mathbf{g}_n is given by $\mathbf{z}_n = \mathbf{z}(\mathbf{g}_n)$, therefore the matrix Z is given by:

$$Z = [\mathbf{z}_1 \quad \mathbf{z}_2 \quad \dots \quad \mathbf{z}_N]^T$$

The n -th ξ term is given as ξ_{ni} .

Outputs of the network given hidden layer vector \mathbf{g}_n is given by $\mathbf{y}_n = \mathbf{y}(\mathbf{g}_n)$, therefore the Matrix Y is given by:

$$Y = [\mathbf{y}_1 \quad \mathbf{y}_2 \quad \dots \quad \mathbf{y}_N]^T$$

The n -th teacher signal is given as t_n and the n -th ρ term is given as ρ_{nk} .

We adjust the weights of the AutoEncoder to minimize the mean squared error (MSE) on training set which is given via the form ℓ_1 in equation 2.5:

$$\ell_1 = \frac{1}{2} \sum_{n=1}^N \sum_{i=1}^I (x_{ni} - z_{ni})^2 \quad (2.5)$$

The cost function of the MLP is given via cross entropy as ℓ_2 and its shown in equation 2.6:

$$\ell_2 = - \sum_{n=1}^N \sum_{h=1}^K t_{nh} \log y_{nh} \quad (2.6)$$

2.3.2 Weight Update Rules

The network learning process of this network contains updating 3 weights: u_{hi} , w_{ih} and v_{kh} . We are interested in minimizing the error function with respect to each of the three weights.

Update rule for u_{hi}

While the error back-propagates from both outputs (AutoEncoder and MLP), the overall error function is given in equation 2.7:

$$L = \ell_1 + \lambda \ell_2 \quad (2.7)$$

therefore minimizing the error function with respect to the weight u_{hi} is given in equation 2.8:

$$\frac{\partial L}{\partial u_{hi}} = \sum_{n=1}^N \omega_{nh} \hat{\delta}_{nwh} x_{ni} + \lambda \omega_{nh} \hat{\delta}_{nvh} x_{ni} \quad (2.8)$$

Given that the term $\hat{\delta}_{nwh} = \sum_{i=1}^I (x_{ni} - z_{ni}) w_{ih}$ and $\hat{\delta}_{nvh} = \sum_{k=1}^K (y_{nk} - t_{nk}) v_{kh}$ that is the weighted delta and $\omega_{nh} = g_{nh}(1 - g_{nh})$. The updated weight due to the back propagation is given in equation 2.9:

$$\Delta u_{hi} = -\alpha \frac{\partial L}{\partial u_{hi}} \quad (2.9)$$

were α is the learning rate, the equation 2.9 leads to the following conclusion:

$$u_{hi_{New}} = u_{hi_{old}} - \alpha \sum_{n=1}^N \omega_{nh} x_{ni} (\hat{\delta}_{nwh} + \lambda \hat{\delta}_{nvh}) \quad (2.10)$$

Update rule for w_{ih}

This time we are interested in minimizing the error function with respect to the weight w_{ih} , Similarly the updated weight due to the back propagation is given by equation 2.11:

$$w_{ih_{New}} = w_{ih_{old}} - \alpha \sum_{n=1}^N \delta_{ni} g_{nh} \quad (2.11)$$

Where $(x_{ni} - z_{ni})$ is the delta term, referenced as δ_{ni} .

Update rule for v_{kh}

While the error back-propagates from both outputs (AutoEncoder and MLP), Given that $\delta_{nk} = (y_{nk} - t_{nk})$ is the delta term, the updated weight due to the back propagation is given by equation 2.12:

$$v_{kh_{New}} = v_{kh_{old}} - \lambda \sum_{n=1}^N \delta_{nk} g_{nh} \quad (2.12)$$

2.3.3 Additive noise -Hidden layer

The simplest form of neural network contamination can be evaluated by injecting noises to hidden units in Multilayer Perceptron (MLP) with one hidden layer [44]. Observing the expectation of the partial derivatives of the squared error with respect to the weights shows the weights from the hidden units to the output units tend to get smaller. On the other side the outputs of the hidden units tend to be 0 or 1 [24]. We extend this basic form to deep neural network as shown in figure 2.

Adding noise to the hidden layer contaminates the units by identical independent Gaussian noise with zero mean and variance of δ^2 .

Denote ϵ_h be the noise added to η_h , and the contaminated vector $\tilde{\boldsymbol{\eta}} = [\eta_1 + \epsilon_1 \quad \cdots \quad \eta_H + \epsilon_H]^T$, $\boldsymbol{\epsilon} = [\epsilon_1 \quad \cdots \quad \epsilon_H]^T$ and the signal transmitted to hidden unit is given by equation 2.13:

$$\tilde{\eta}_h = \mathbf{u}_h^T \mathbf{x} + q_h + \epsilon_h = \eta_h + \epsilon_h \quad (2.13)$$

Given $\omega_{nh} = g_{nh}(1 - g_{nh})$ and $\nu_h = g_h(1 - g_h)(1/2 - g_h)$, the term \tilde{g}_{nh} is given as , the output vector $\tilde{\boldsymbol{g}}$ is given by Taylor expansion as in equation 2.14:

$$\tilde{g}_h \approx g_h + \omega_{nh}\epsilon_h + \nu_{nh}\epsilon_h^2 \quad (2.14)$$

The i-th output of the AutoEncoder \tilde{z}_{ni} now becomes as shown in equation 2.15:

$$\begin{aligned} \tilde{z}_{ni} &\approx \mathbf{w}_i^T \tilde{\boldsymbol{g}}_n + r_i \\ &= z_{ni} + \sum_{h=1}^H w_{ih}(\omega_{nh}\epsilon_h + \nu_{nh}\epsilon_h^2) = z_{ni} + \Delta z_{ni} \end{aligned} \quad (2.15)$$

On the other hand, the output units of the network are also affected by the noise injection. This is represented in equation 2.16:

$$\begin{aligned}\tilde{\rho}_k &\approx \mathbf{v}_k^T \tilde{\mathbf{g}} + s_k \\ &= \rho_k + \sum_{h=1}^H v_{kh}(\omega_h \epsilon_h + \nu_h \epsilon_h^2) = \rho_k + \Delta \rho_k,\end{aligned}\quad (2.16)$$

and

$$\tilde{y}_k = \frac{e^{\tilde{\rho}_k}}{\sum_{i=1}^K e^{\tilde{\rho}_i}} \quad (2.17)$$

Update rule for u_{hi}

We are interested in minimizing the error function with respect to the weight u_{hi} . To denote the loss function with noise injection, we denote $\tilde{\ell}_1 = \frac{1}{2} \sum_{n=1}^N \sum_{i=1}^I (\tilde{z}_{ni} - z_{ni})^2$ for AutoEncoder,

and $\tilde{\ell}_2 = - \sum_{n=1}^N \sum_{k=1}^K t_k \log \tilde{y}_k$ for the softmax classifier, given that $\tilde{L} = \tilde{\ell}_1 + \tilde{\ell}_2$ then

$$\begin{aligned}\frac{\partial \tilde{L}}{\partial u_{hi}} &= \sum_{n=1}^N \omega_{nh} \hat{\delta}_{nwh} x_{ni} + \sum_{n=1}^N \sum_{j=1}^I w_{jh} \Delta z_{nj} \omega_{nh} x_{ni} \\ &+ \sum_{n=1}^N \sum_{j=1}^I \left((1 - 2g_{nh}) \epsilon_h + \frac{1}{2} \chi_{nh} \epsilon_h^2 \right) w_{jh} \delta_{nj} \omega_{nh} x_{ni} \\ &+ \sum_{n=1}^N \sum_{j=1}^I 2w_{jh}^2 \epsilon_h^2 \omega_{nh} \nu_{nh} x_{ni} \\ &+ \sum_{n=1}^N \sum_{k=1}^K \tilde{\delta}_{nk} \omega_{nh} x_{ni} \left(1 + \frac{\chi_{nh} \epsilon_h^2}{2} \right).\end{aligned}\quad (2.18)$$

where $\chi_{nh} = 1 - 6g_{nh} + 6g_{nh}^2$. The detailed derivation of equation 2.18 can be found at the appendix.

The expectation value of $\frac{\partial \tilde{L}}{\partial u_{hi}}$ is given in equation 2.19:

$$\mathbb{E} \left[\frac{\partial \tilde{L}}{\partial u_{hi}} \right] = \mathbb{E} \left[\frac{\partial \tilde{\ell}_1}{\partial u_{hi}} \right] + \mathbb{E} \left[\frac{\lambda \partial \tilde{\ell}_2}{\partial u_{hi}} \right] \quad (2.19)$$

Where $\mathbb{E} \left[\frac{\partial \tilde{\ell}_1}{\partial u_{hi}} \right]$ equals to:

$$\begin{aligned} &= \frac{\partial \tilde{\ell}_1}{\partial u_{hi}} \left(1 + \frac{\chi_{nh} \sigma^2}{2} \right) + 3 \sum_{n=1}^N \sum_{j=1}^I w_{jh}^2 \omega_{nh} \nu_{nh} x_{ni} \sigma^2 \\ &\quad + \sum_{n=1}^N \sum_{j=1}^I w_{jh} \omega_{nh} x_{ni} \sigma^2 \left(\sum_{h'=1, h' \neq h}^H w_{jh'} \nu_{nh'} \right) \end{aligned} \quad (2.20)$$

From equation 2.20, we can see that the term $\chi_{nh} \sigma^2 / 2$ takes negative values when the output g_{nh} is in the range $1/2 - \sqrt{3}/6 < g_{nh} < 1/2 + \sqrt{3}/6$ while it takes positive values when the output g_{nh} approaches to 0 or 1. Therefore, the learning rate is suppressed when y is near $1/2$, an uncertain value, and is accelerated when g_{nh} is near 0 or 1, a definite value.

The expectation value of $\frac{\partial \tilde{\ell}_2}{\partial u_{hi}}$ is estimated by Jensen's inequality as shown in equation 2.21:

$$\begin{aligned} \mathbb{E} \left[\log \sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}} \right] &\leq \log \left(\mathbb{E} \left[\sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}} \right] \right) \\ &= \log \left(\sum_{n=1}^N \sum_{j=1}^K e^{\rho_{nj} + \text{Var}[\tilde{\rho}_{nj}]/2} \right) \end{aligned} \quad (2.21)$$

where $\text{Var}[\tilde{\rho}_{nj}] \approx \sum_{i=1}^H (v_{ji} \omega_{ni} \sigma)^2$. Therefore, we have

$$\begin{aligned} \mathbb{E}[\log \tilde{y}_{nk}] &= \rho_{nk} \\ &\quad + \left(\sum_{h=1}^H v_{kh} \nu_{nh} \right) \sigma^2 - \mathbb{E} \left[\log \sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}} \right] \\ &\geq \rho_{nk} + \left(\sum_{h=1}^H v_{kh} \nu_{nh} \right) \sigma^2 - \log \left(\sum_{n=1}^N \sum_{j=1}^K e^{\rho_{nj} + \text{Var}[\tilde{\rho}_{nj}]/2} \right). \end{aligned} \quad (2.22)$$

The detailed derivation of equation 2.21 and 2.22 can be found at the appendix.

From equation 2.22, we can see that the term v_{nh} is close to 0 when the output g_{nh} approaches to 0 or 1, thus makes the second term be closer to 0 in this case. Similarly, when g_{nh} is close to 0 or 1, ω_{nh} is also close to 0, and makes the variance $Var[\tilde{\rho}_{nj}]$ be closer to 0. In summary, when g_{nh} becomes a definite value, $\mathbb{E}[\log \tilde{y}_{nk}]$ is close to ρ_{nk} and that makes $\mathbb{E}[\tilde{y}_{nk}]$ is close to y_{nk} . Since χ_{nh} is positive when g_{nh} is close to 0 or 1, we have

$$\mathbb{E} \left[(\tilde{y}_{nk} - t_{nk}) \frac{\partial \rho_{nk}}{\partial u_{hi}} \left(1 + \frac{\chi_{nh} \epsilon_h^2}{2} \right) \right] \geq (y_{nk} - t_{nk}) \frac{\partial \rho_{nk}}{\partial u_{hi}} \quad (2.23)$$

When the outputs of hidden layer units are uncertain, $g_{nh} \approx 1/2$ for all h , the gap between two sides of Jensen's inequality is almost zero. And also, $v_{nh} = 0$ when $g_{nh} = 1/2$. Therefore in equation 2.24 we have

$$\log \mathbb{E}[\tilde{y}_{nk}] \approx \rho_{nk} - \log \left(\sum_{n=1}^N \sum_{j=1}^K e^{\rho_{nj} + Var[\tilde{\rho}_{nj}]/2} \right). \quad (2.24)$$

In this case we have $\log \mathbb{E}[\tilde{y}_{nk}] \leq \rho_{nk}$, which implies that $\mathbb{E}[\tilde{y}_{nk}] \leq y_{nk}$. Since $\chi_{nh} < 0$ when g_{nh} is close to 1/2, we can conclude that the learning rate is suppressed.

Update rule for v_{kh}

the expectation vale of $\frac{\partial \tilde{\ell}_2}{\partial v_{kh}}$ is given in 2.25:

$$\mathbb{E} \left[\frac{\partial \tilde{\ell}_2}{\partial v_{kh}} \right] \approx \frac{\partial \ell_2}{\partial v_{kh}} + (y_{nk} - t_{nk}) v_{nh} \sigma^2. \quad (2.25)$$

Again, when g_{nh} takes definite value 0 or 1, we have $\mathbb{E}[\tilde{y}_{nk} - t_{nk}] \approx \mathbb{E}[y_{nk} - t_{nk}]$ and when g_{nh} becomes a definite value, $\mathbb{E}[\log \tilde{y}_{nk}]$ is close to ρ_{nk} and that makes $\mathbb{E}[\tilde{y}_{nk}]$ is close to y_{nk} which enhances the learning process.

2.3.4 Additive noise -Input layer

In this section we discuss the contamination of all input features by identical independent Gaussian noise with zero mean and variance of σ^2 . The contaminated vector $\tilde{\mathbf{x}} = [x +$

$\varepsilon_1 \ \cdots \ x_I + \varepsilon_I]^T$, $\boldsymbol{\varepsilon} = [\varepsilon_1 \ \cdots \ \varepsilon_I]^T$ and the signal transmitted to hidden unit is

$$\tilde{\eta}_h = \mathbf{u}_h^T \tilde{\mathbf{x}} + q_h = \mathbf{u}_h^T \mathbf{x} + \mathbf{u}_h^T \boldsymbol{\varepsilon} + q_h := \eta_h + \epsilon_h,$$

where $\epsilon_h := \mathbf{u}_h^T \boldsymbol{\varepsilon}$, similarly we obtain the expectation values of gradients referred to as

$$\begin{aligned} & \mathbb{E} \left[\frac{\partial \tilde{\ell}_1}{\partial u_{hi}} \right] \text{ and } \mathbb{E} \left[\frac{\partial \tilde{\ell}_2}{\partial u_{hi}} \right] \\ & \mathbb{E} \left[\frac{\partial \tilde{\ell}_1}{\partial u_{hi}} \right] \\ & = \frac{\partial \ell_1}{\partial u_{hi}} \left(1 + \frac{\chi_{nh} \|\mathbf{u}_h\|_2^2 \sigma^2}{2} \right) \\ & + \sum_{n=1}^N \sum_{j=1}^I w_{jh} \omega_{nh} x_{ni} \sigma^2 \left(\sum_{h'=1, h' \neq h}^H w_{jh'} v_{nh'} \|\mathbf{u}_{h'}\|_2^2 \right) \\ & + 3 \sum_{n=1}^N \sum_{j=1}^I w_{jh}^2 \omega_{nh} v_{nh} x_{ni} \sigma^2 \|\mathbf{u}_h\|_2^2 \\ & + \sum_{j=1}^I w_{jh}^2 \omega_{nh}^2 x_{ni} \sigma^2 \end{aligned} \quad (2.26)$$

From equation 2.26, we can see that the term $\chi_{nh} \sigma^2 \|\mathbf{u}_h\|_2^2 / 2$ takes negative values when the output g_{nh} is in the range $1/2 - \sqrt{3}/6 < g_{nh} < 1/2 + \sqrt{3}/6$ while it takes positive values when the output g_{nh} approaches to 0 or 1. Therefore, the learning rate is suppressed when y is near 1/2, an uncertain value, and is accelerated when g_{nh} is near 0 or 1, a definite value.

The expectation value $\mathbb{E} \left[\frac{\partial \tilde{\ell}_2}{\partial u_{hi}} \right]$ is described in equation 2.26:

$$\begin{aligned} & \mathbb{E} \left[\frac{\partial \tilde{\ell}_2}{\partial u_{hi}} \right] = \\ & \left(\sum_{n=1}^N \sum_{k=1}^K v_{kh} \mathbb{E}[\tilde{y}_{nk} - t_{nk}] \right) \omega_{nh} x_{ni} \left(1 + \frac{\chi_{nh} \sigma^2 \|\mathbf{u}_h\|_2^2}{2} \right) \\ & + 2 \left(\sum_{n=1}^N \sum_{k=1}^K \mathbb{E}[(\tilde{y}_{nk} - t_{nk}) \varepsilon_i^2] v_{kh} v_{nh} u_{hi} \right) \end{aligned} \quad (2.27)$$

Recall that

$$\tilde{\rho}_{nk} = \frac{e^{\tilde{\rho}_{nk}}}{\sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}}} \quad (2.28)$$

The softmax operator is to normalize the values $e^{\tilde{\rho}_{nk}}$ in the output units. We now analyze the influence of noise injection on $e^{\tilde{\rho}_{nk}}$ through equations 2.29 and 2.30.

$$e^{\tilde{\rho}_{nk}} = e^{\rho_{nk}} \cdot e^{\sum_{h=1}^H v_{kh} \omega_{nh} \epsilon_h} \cdot e^{\sum_{h=1}^H v_{kh} v_{nh} \epsilon_h^2} \quad (2.29)$$

$$\mathbb{E}[e^{\tilde{\rho}_{nk}}] = e^{\rho_{nk}} \mathbb{E}[e^{\sum_{h=1}^H v_{kh} \omega_{nh} \epsilon_h} \cdot e^{\sum_{h=1}^H v_{kh} v_{nh} \epsilon_h^2}] \quad (2.30)$$

When the variance of noise is small, the amplitude of ϵ_h^2 can be negligible. In this case, we can simplify the expectation $\mathbb{E}[e^{\tilde{\rho}_{nk}}]$ as shown in equation 2.31:

$$\mathbb{E}[e^{\tilde{\rho}_{nk}}] = e^{\rho_{nk} + \sum_{h=1}^H v_{kh}^2 \omega_{nh}^2 \sigma^2 \|\mathbf{u}_h\|_2^2 / 2} \quad (2.31)$$

Since the term $\sum_{h=1}^H v_{kh}^2 \omega_{nh}^2 \sigma^2 \|\mathbf{u}_h\|_2^2$ is always nonnegative, it is clear that $\mathbb{E}[e^{\tilde{\rho}_{nk}}] \geq e^{\rho_{nk}}$. Thus, the normalized value \tilde{y}_{nk} will also be increased when noise is injected. The variance value of noise is small when g_{nh} is close to 0 or 1, therefore the normalized value \tilde{y}_{nk} will be increased when noise is injected. Since the term $v_{nh} \sigma^2 \|\mathbf{u}_h\|_2^2 > 0$, we can conclude that the learning is accelerated when updating \tilde{y}_{nk} by injecting slight noise.

Update rule for v_{kh}

Now we proceed to find updating rules for v_{kh} . Its gradient expectation is given by equation 2.32:

$$\mathbb{E} \left[\frac{\partial \tilde{\ell}_2}{\partial v_{kh}} \right] = v_{nh} \mathbb{E}[\epsilon_h^2] = v_{nh} \sigma^2 \|\mathbf{u}_h\|_2^2 \quad (2.32)$$

From above analysis, we can see that when the variance of noise is small, the normalized value \tilde{y}_{nk} will be increased when noise is injected. Since the term $v_{nh} \sigma^2 \|\mathbf{u}_h\|_2^2 > 0$, we can conclude that the learning is accelerated when updating \tilde{y}_{nk} by injecting slight noise.

2.4 Experiment

In our experiment we train and evaluate classification networks using MNIST dataset[13]. Initially we only use the MLP to conduct the basic classification experiment. Later in our experiment we consider 2 cases, stacked and non-stacked AutoEncoder, each is connected to the MLP. Two stages per case is considered: noiseless and noisy training. In all experiments, Images with 28 by 28 pixels per is used which defines the input size of the network. we compare the network accuracy and overall loss for every case.

Basic case: Classifying with an MLP

Logistic regression is a linear classifier. It is described in terms of its weight matrix and a bias vector. Having certain number of classes, the classification is performed by projecting an input vector onto a set of planes, each of which corresponds to a class. The Learning process aims to build an optimal model parameters. This process involves minimizing a loss function. In the case of multi-class logistic regression. A common loss function is the negative log-likelihood. The evaluation criteria of this experiment is the accuracy of classification and the overall loss evaluated by loss functions. We used 16,000 MNIST images, 6,000 for training and 10,000 for testing. The training took 1,000 Epochs with mini-batch size of 100. The loss function is the cross entropy, during this step we evaluated noiseless training and training with noise. The classification detection rate was 98.00% while the loss was 0.0023 with noise. On the other hand classification detection rate was 97.60% while the loss was 0.003 without noise. Table 1 shows the basic case results in its last 2 rows.

Nested case: Classifying with AutoEncoder and an MLP

Secondly an AutoEncoder is added to the MLP. The experiment used 16,000 images of MNIST, 6,000 of those images were used for training and 10,000 images for testing. The training took 1,000 Epochs with mini-batch size of 100. The loss function $(\lambda)CE + (1 - \lambda)MSR$ is balanced by $\lambda = 0.6$ based on multiple experiments. ADAM optimizer is used for the training without weight decays. Adaptive Moment Estimation (Adam) [25] is one of the techniques to find adaptive learning rates for each parameter within the network. Figure 4.1 shows a rough description for the network and how the AutoEncoder hidden units is directly connected to MLP at one case and stacked with another AutoEncoder in the second case. The number of units in a AutoEncoder hidden layer is varied to cover 10, 50, 100, 500, 700 and 1,000 hidden units. Increasing the hidden units increases the accuracy and lowers the loss. At this stage noise injection is not enabled.

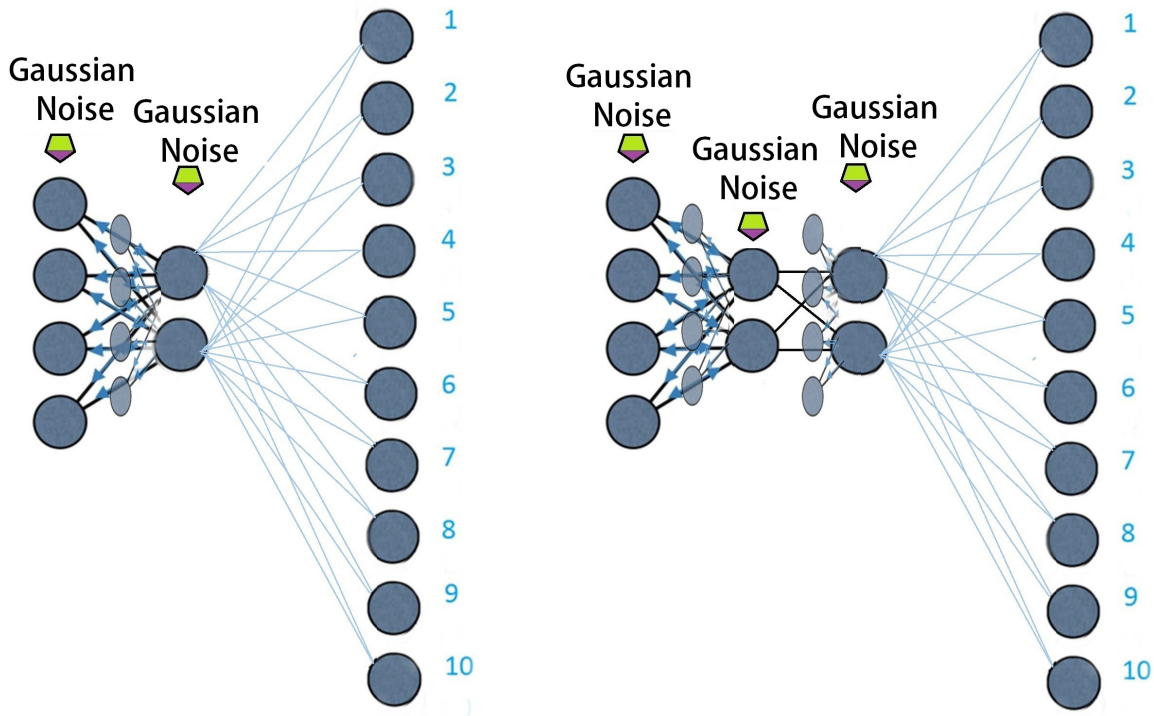


Fig. 2.2 The network to the left shows a single AutoEncoder connected to an MLP through its hidden layer. To the right is the network that stacks AutoEncoders and connect its hidden units to a MLP. The arrows at the input layer and hidden layers are noise injection joints.

On the other hand stacking the AutoEncoder has useful effects on the network performance. It tends to learn features like edges in an image. Such features are called first-order features. The second layer of a stacked AutoEncoder has the ability to understand more features detected from the first layer such as which edges tend to occur together to form contour or corner detectors. Considering the modules we discussed we designed a multi-layer AutoEncoder that is connected to MLP through its hidden units. The effect of increasing the hidden units is similar to the single layered AutoEncoder. Increasing the hidden units increases the accuracy, lowers the loss.

In both cases the number of hidden units per layer is varied, the accuracy of classification and the loss are evaluated. Less number of hidden units tend to give less accuracy and higher loss. This is natural while the ability to capture features becomes limited.

Noisy case: contaminating Nested Network

To confirm the effectiveness of noise contamination we inject the Gaussian noise which comes with a mean of 0 and the standard deviation $\sigma = 0.1$ based on multiple experiments into hidden and input units of AutoEncoder (Single layered and stacked) while being attached

to an MLP. Table 1 shows the variation of noise states for a 1,000 hidden-units single AutoEncoder linked to MLP network. Increasing hidden units through noise injection improves the performance as figure 2.3 illustrates.

Figure 4.1 shows the contamination points. increasing the hidden units number has similar effect to the noiseless case as figure 2.4 illustrates. Injecting noise to the hidden layer again shows the best performance. Table 2 puts more light on the 1,000 hidden units noise injection for the stacked AutoEncoder case. This indeed matches the conclusion of equations derived in section 3.

Table 1 -

Changing the noise state has significant effect at the network accuracy and loss. Injecting noise at the hidden layer of the AutoEncoder outperforms all the other cases.

Dsign	Noise Position	Accuracy (Train)	Accuracy (Test)	Loss (Train)	Loss (Test)
AE + MLP	None	84.79% (± 3.23)	85.62% (± 4.01)	0.0092	0.0079
AE + MLP	Input	86.96% (± 2.96)	88.20% (± 2.59)	0.0082	0.0067
AE + MLP	Hidden	97.09% (± 2.54)	98.51% (± 1.36)	0.00023	0.0001
AE + MLP	Input + Hidden	90.91% (± 2.3)	91.39% (± 1.52)	0.0084	0.0069
MLP	None	96.25% (± 2.96)	97.60% (± 1.98)	0.0030	0.003
MLP	Hidden	97.08% (± 1.03)	98.00% (± 0.76)	0.0031	0.003

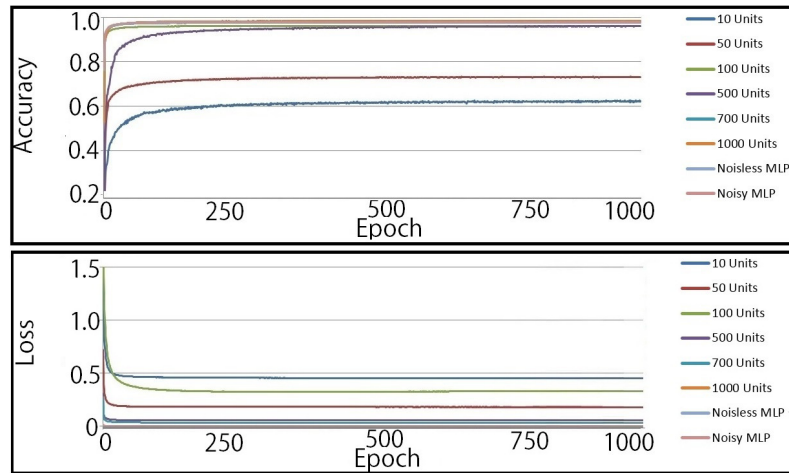


Fig. 2.3 -

Injecting noise into the hidden units of the single layered AutoEncoder improves the accuracy (top) and reduces the loss (bottom). The effect becomes clearer during the increase of hidden units number.

Table 2 -

Networks with stacked AutoEncoders have better ability to extract important features. Injecting noise within their hidden units clearly improves the performance. A 1000 hidden unit stacked AutoEncoders is used and noise injection results are shown.

Dsign	Noise Position	Accuracy (Train)	Accuracy (Test)	Loss (Train)	Loss (Test)
AE + MLP	None	85.22% (± 7.26)	86.14% (± 5.01)	0.0056	0.005
AE + MLP	Input	89.52% (± 3.56)	91.70% (± 2.55)	0.0042	0.0037
AE + MLP	Hidden	98.09% (± 0.29)	99.65% (± 0.06)	0.000039	0.000025
AE + MLP	Input and Hidden	93.52% (± 2.5)	94.00% (± 1.9)	0.0056	0.004

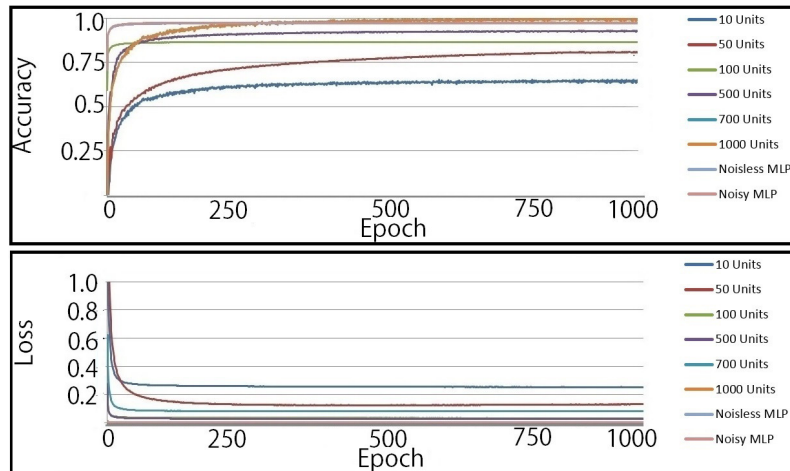


Fig. 2.4 noise within the hidden units of and MLP with stack layered AutoEncoder improves the accuracy (top) and reduces the loss (bottom).The effect becomes clearer during the increase of hidden units number.

To add experimental understanding to our work, AutoEncoder's hidden layer output is visualized for noisy and noiseless schemes under hidden units number variation. We observe stronger feature representation when noise is injected into any network size and depth. Figure 2.5 shows how the internal representation of the features improves due to noise injection.

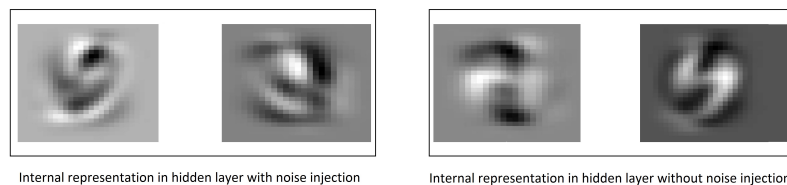


Fig. 2.5 AutoEncoder often captures a useful "hierarchical grouping" of the input. Visualizations for the hidden layer output shows stronger features of the digits after noise injection (left) compared to noiseless case (right).

On the same hand the sparsity of hidden units activation increases as a measure of better feature representation as shown in the histogram chart in figure 2.6.

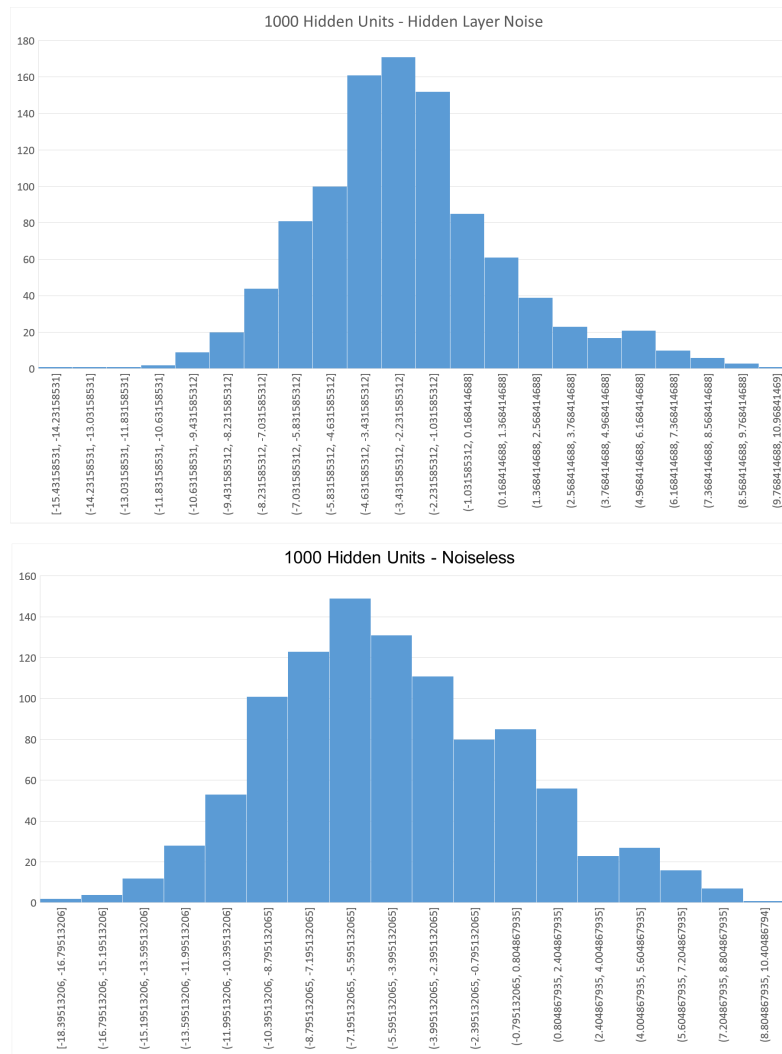
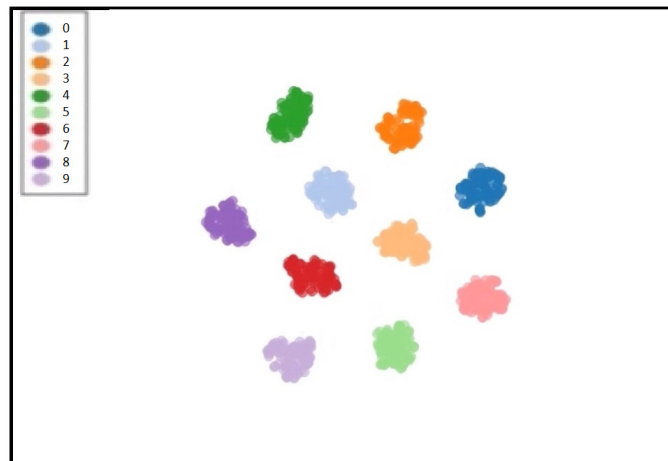


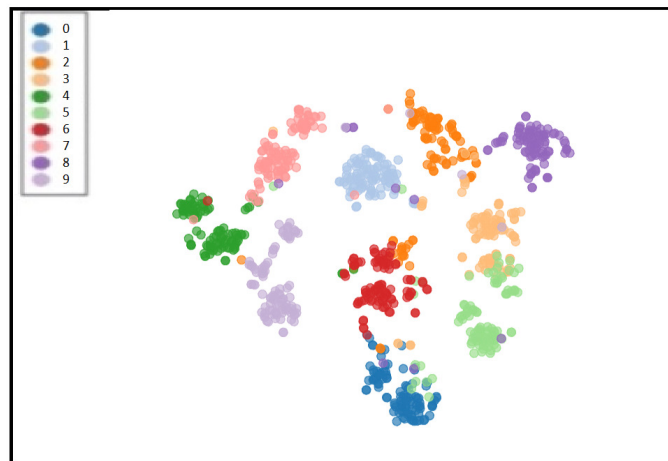
Fig. 2.6 Upon giving certain input to the network the activation of hidden units has higher density around certain value that represents better sparsity.

Figure 2.7 illustrates sparsity through the "t-SNE" [26] diagram that visualizes high-dimensional data by giving each hidden unit activation a location in a two dimensional map.

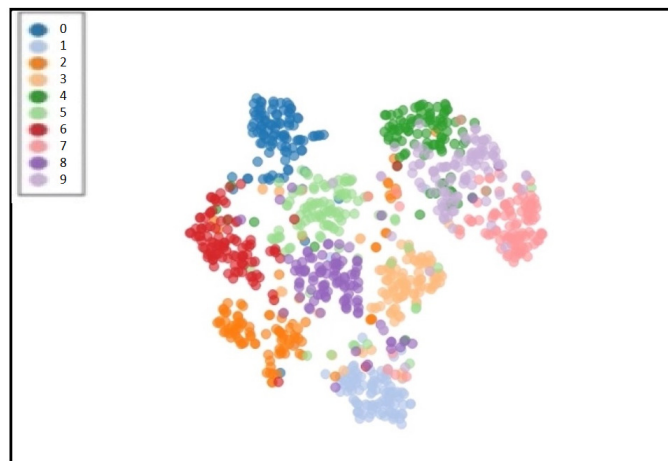
The t-SNE shows the hidden unit activation across all input cases. In the noiseless case there is considerable overlap while in noisy training case the overlap is almost zero which gives another measure of how noise injection improves the network performance.



TSNE representation -Hidden layer noise injection



TSNE representation - Input layer noise injection



TSNE representation without noise injection

Fig. 2.7 -

The t-SNE clearly shows better sparsity in case of noisy training(top) while inputs representations in hidden layer are clearly not overlapping. On the other hand the t-SNE chart shows more overlapping in the case of injecting noise at the input(middle). Finally overlapping among activation is the highest when noiseless training is performed for the same network(bottom).

2.5 Conclusion

In this chapter, we investigated the effect of injecting noise on enhancing learning and improving generalization of neural networks made of AutoEncoders connected to MLP. We proposed mathematical clarification on why injecting noise at certain locations increases the network performance. Experimentally, injecting noise at those locations is able to obtain a better performance for both deep and shallow networks. The proposed method was demonstrated through classification problem of digits under variety of hand writing. The experimental findings match our hypothesis which is supported by mathematical evidence.

The next chapter is followed with more emotional oriented neural network model that investigate feature localization abilities upon injecting noise into the convolutional neural network. A model that classifies the 7 human emotional states based on facial expressions is proposed and it is shown to perform better than the earlier convolutional neural networks.

Chapter 3

Enhancing emotions estimation accuracy by additive noise

3.1 Background

Beside speech and body language, emotions play an important role in our daily routines of information exchange. Interpreting emotions enhances our understanding of each other and deepens our conversations. Extending emotion understanding to computer field enriches the human computer interaction and allows more natural ways of collaboration.

Commonly, when describing an emotion, facial expressions are considered. Many algorithms rely on extracting important facial regions to predict an emotion. Facial Action Coding System (FACS) [27] is basis for many recent emotion recognition algorithms. It's a set of facial actions which defines an emotion such as contracting glabella when being angry. Such facial actions define the facial expression and therefore an emotional state [28]. In this chapter, we propose a Convolutional Neural Networks (CNN) based method to classify emotions from the facial feature.

Many neural network models have been introduced to analyze facial features [29–31]. CNN's have proven to be good features extractors for such task. Their ability to observe input through different filters widens their scope of operation. They can also localize the discriminative regions for action classification instead of classifying only objects [129, 145]. Feature localization helps understanding reasons of misclassification and the way knowledge is built within the network. GoogleNet [34] is a promising architecture that relies on CNNs to extract features through multiple perspectives. It achieves an object detection accuracy of 44% in ILSVRC challenge [35]. Such high precision has dragged researchers' attention to this architecture. The main component of GoogleNet is called the Inception model. Its architecture emulates the animal visual cortex [37]. The Inception modules are blocks that consist of Convolutional, Pooling, and RELU Layers. It can find optimal local sparse structure in a convolutional vision network due to its dual feedforward paths. We find this

model constructive in our proposed neural network for facial feature extraction. To the best of our knowledge, it hasn't been used in facial feature extractions analysis before. We stacked two Inceptions blocks to extract more silent features that are used for classification.

We suppress the number of effective weights by noise injection as a regularizer. Injecting noise directs the learning algorithm towards a solution with the least possible amount of zero weights [36, 42, 41, 43]. Kurita et al. [44] analyzed the backpropagation learning behavior while injecting noise into the inputs of the hidden units during simple MLP training. In their experiment, they showed that noisy training gives the network ability to automatically structure itself. They also observed that the links from hidden layer to output layer hold smaller weights while the links from the input layer to hidden layer holds larger weights values. Sabri et al. [43] extended Kurita's work using a deeper network model showing how the internal representations of the networks benefit from noise injection. Kartik et al. [45] analyzed same technique on CNN which generates a hyperplane in noise space by maximizing noisy expectation. They showed that injecting noise into neural network boosts the backpropagation training of a neural network.

Motivated by the need to expand neural network generalization abilities, we analyze injecting Gaussian noise selectively to multiple joints within a deep CNN and its relation with feature localization and performance. This is conducted through emotion classification experiments based on facial expressions. We generate activation maps and feature maps to visualize the internal representation of the network. Our experiments steadily result in accuracy improvements and clearer localization abilities of important features (e.g. higher activation around happiness facial features (Mouth and cheeks) and stressing facial features (glabella)) upon injecting Gaussian noise into selected joints of the neural network to classify the 7-emotional states in the KDEF dataset [46] and CK+ dataset [141]. Our experiments used a stacked Inception models of GoogleNet [34] that we connected to a classifier to classify KDEF portrait images into 7-emotional states. This matches the finding of recent researches that evaluate internal representations change upon regularizing the training of a network [43]. Critical facial parts have higher activations while other regions have lower values when compared to the noiseless case.

This chapter is organized as follows. In section 2, relevant models and noise injection techniques are reviewed as related work. The details of the approach used in this chapter are explained in section 3. The model definition for noiseless and noisy training is mentioned in section 3.A and 3.B respectively. Feature visualization details for both feature maps and class activation maps are mentioned in section 3.C. Experiments details are described in section 4. Discussion of the results is in section 5. Finally, the conclusion is shown in section 6.

3.2 Related work

CNN's efficiency in various recognition tasks has dragged many research attentions. Many structures have been used in the field of emotion classification [39, 40]. Facial expressions are considered key as elements when analyzing an emotional state but a challenge as well due to the variety of sizes of facial components for different subjects. In this work, we propose CNN architecture inspired by GoogleNet Inception model that classifies emotional states based on facial expressions. We also show that injecting noise during the training improves the network classification and feature localization abilities. Here, we discuss the work most related to this chapter: Inception model, regularizing neural networks by injecting noise.

3.2.1 Inception model

Szegedy et al. [34] proposed Inception model as part of GoogleNet architecture. It consists of Convolutional, Pooling, and RELU Layers. This architecture abstracts features from different scales and allows dimension reduction before convolving over filters output with a large patch size. Inception modules optimizes sparse structure through searching for a local minimum in certain domains. The Inception block first reduces the dimension since it convolves with a kernel of small size. Rectified linear unit (RELU) is used to enforce generalization and suppress ambiguity. The RELU output is then forwarded to convolution layer with bigger kernel size. The same data that was given to the Inception modules has its dimension reduced by max pooling and then forward to convolution layer. The different kernel sizes support multiple analysis scale and results in numerous output filter banks that is concatenated into a single vector forming Inception output.

Those characteristics allowed many researchers to use Inception model in classification problems for features with different sizes. This is found to be essential when it comes to facial expression interpretation. Regions like lips, eye corners and eyebrows shapes are key local regions with high sparse representations. YY et al. [49] used Inception model in face recognition using stacked convolution and Inception layers. They recognized subjects faces regardless the distance of the subject from the camera. Florian et al. [50] used Inception model to learn mapping from face images to a compact Euclidean space to find similarities considering multiple subject facial alignments. Their method optimizes the features themselves instead of the intermediate bottleneck layer. Introducing the parallel paths inspired by inspection model overcomes size diversity problem of the traced features. A similar scheme was proposed for people age and gender classification [51].

3.2.2 Regularization: Noise injection

The nature of data has been always a challenge when training CNN's. Altering the training process of a deep neural network improves the performance of learning process via numerous techniques [52]. Gaussian Noise injection to hidden units has been utilized in many neural networks researches for many years [53, 54, 38]. Kurita et al. [44] added noise into the hidden layers of an MLP resulting in automatic structurization which improves the generalization ability of the network. Adding randomness improves the ability to escape local minima or passing through the early training phase in the most optimum phase. Zeyer et al. [54] showed that adding gradient noise avoids over-fitting and result in lower training loss. Sabri et al.[43] extended Kurita et al. [44] work by analyzing the performance of deeper network structures upon injecting noise into certain locations. Their extended mathematical and experimental work matches the findings of Kurita et al. [44].

3.2.3 Emotion recognition by using CNN

In this chapter, we extend the usage of Inception model towards emotion analysis. Recent CNN architectures offer significant improvements in understanding facial expressions. Song et al. [55] utilized a deep CNN for learning facial expressions. They used multiple layers of CNN and local filter layers and achieved an accuracy of 99.2% on the Extended Cohn-Kanade Dataset [56]. When using similar architectures with a more challenging dataset, classification accuracy becomes lower. Kukla et al. [58] used two CNN models to classify emotions in KDEP [46] dataset. Their work is represented in a confusion matrix with values vary between 54% and 92%. For the same dataset Alessandro et al.[57] using cascade networks having results confusion matrix with values between 76% and 94%. Kuilenburg et al.[59] showed results between 80% and 97% for the same dataset. Garcia et. al [48] compared the performance of Support Vector Machines (SVM) and a Multilayer Perceptron Network (MLP) on facial expression classification. They cropped irrelevant spatial features and applied Gabor filters to images as pre-processing step before classification. Their SVM proposed model with accuracy rate of 97.08 % overcomes MLP approach which has an accuracy rate of 93.5%.

It is interesting to understand the nature of features extraction that leads to those results. Visualizing how a network uses the extracted features to build knowledge is key to understand why misclassification occurs. Visualization can be for network weights or attention regions. Ren et al. [60] provide insights about the region-wise visualization. They showed that the position of convolution sliding window provides localization information regarding the image. This resulted in maps that they refer to as feature maps. Zhou et al. [129] proposed

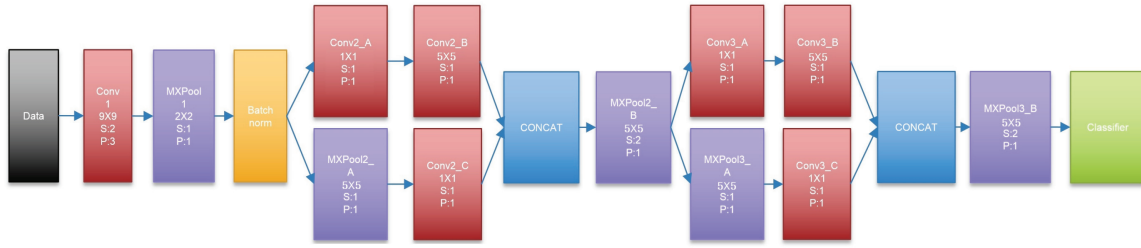


Fig. 3.1 The proposed architecture shows the two Inception blocks stacked and connected to a SoftMax classifier, the kernel size is shown on every block and S represents the Stride and P represents the Padding.

a method that allows classification-trained CNNs to learn to perform object localization, without using any bounding box annotations. They refer to it as Class Activation Mapping (CAM). CAMs allow visualizing the predicted class scores on any given image, allowing the objects detected by the CNN to be highlighted.

3.3 Proposed Method

In this chapter, we evaluate the effect of injecting noise into multiple joints of CNN connected to a classifier. Our network is trained to classify emotions based on facial expressions as features, the emotions are happiness, surprise, neutral, fear, sadness, disgust and anger. The input data passes through 2 Inception blocks. Each block learns detailed and useful features of the input data, in our case, the KDEF [46] and CK+ [141] databases of people who show emotions through facial expressions [46]. The second block is connected to a classifier that decides which class each input data belongs to depending on a teacher signal.

In order to enable such architecture to realize the diversity of the input, multiple filters can be applied to the input. The filters are realized by sharing weights of neighboring neurons. Comparing with MLP training, CNNs require fewer weights updates during training, since multiple weights are bound together.

3.3.1 Proposed network architecture

Faces that hold the same emotional expressions are classified within the same category based on special features within them. The proposed deep CNN architecture in figure 4.1 has two main blocks stacked and connected to a classifier. The input images are gray scaled in the preprocessing stage. The input is then tunneled to filters of Convolution 1 which analyzes the local regions of the input image and groups into filter banks. We end up with a lot of clusters

concentrated in a single region. The filter's output is then forwarded to the max pooling which down-samples the images and then they are normalized. The normalized output is then forwarded to two stacked Inception blocks. each block extracts more silent features through its parallel path. The detected silent features are forwarded to a classifier to classify them based on a SoftMax loss function.

The structure we used assumes the data is initially gray-scaled in a preprocessing stage. The data enters a 9×9 kernel convolutions with a stride of 2 and padding of 3 followed by a 2×2 max pooling with stride and padding of 1 to down-sample the 64 different filters resulted from the previous convolution. The output is later normalized. The normalized data is then forwarded to a 1×1 convolution kernel with a stride of 1 and padding of 1 followed by 5×5 convolution kernel with a stride of 1 and padding of 1 in the upper path. The upper path reduces the dimension of the input due to the presence of 1×1 convolution. At the same time in a parallel path, the normalized data enters a 5×5 max pooling followed by a 1×1 convolution kernel with a stride of 1 and padding of 1. Both outputs are later concatenated. This concatenation offers more diverse representation of the input. Another identical Inception block receives the concatenated data and its output is forwarded into a 4×4 max pooling with stride of 2 and padding of 1. The max pooling output is forwarded to a 7-classes classifier with SoftMax loss function.

As proposed by GoogleNet[34], kernel sizes are recommended to be between 1×1 , 3×3 and 5×5 . In our work, our selection for kernel sizes reflects the various scales at which facial expressions can appear.

3.3.2 Training with noise

Introducing noise during training showed an improvement in classification tasks [44, 43]. We believe adding noise to the activations of Inception modules hidden layers will improve accuracy and reduce the loss. Unlike standard neural network, convolutional neural network components are more sensitive to noise. Injecting noise into Max pooling or batch normalization units, for instance, disturbs the sole purpose of those units and such interference is considered destructive. In our case, the output of every convolution unit is a candidate for injection. The smallest kernel results in more filters. Therefore, injecting noise at the 1×1 convolutional unit output gives the highest impact.

Contaminating filters output results in similar effects as standard neural network weights contamination. As proposed in [44, 43] learning rate is suppressed when network is uncertain about the current output. This gives less value for unconstructive features. On the other hand, the learning rate is accelerated when the output is more definite. It's also predicted that internal representation of learned features will emerge and sparsity of filter outputs will

increase when independent Gaussian noises are added during the deep network training. This is expected to improve the generalization ability as well of the network through this automatic structuration by adding the noises.

In the next section, we show two schemes for noise injection. Injection at the first Inception block and at the second block. Inception models learn features depending on their location at the network. First Inception block learns features that are called first-order features. The second block can understand more features detected from the first block such as which alignment of facial features or facial features that tend to occur together to form an expression.

In our first noise placement, we add noise between *Conv2_A* and *conv2_B* shown in figure 4.1. the second was conducted while adding noise between *Conv3_A* and *conv3_B*. we evaluated the overall improvement through network classification accuracy, loss and ability of the network to localize facial features more precisely. Increasing the accuracy means the network is looking with more attention towards important features. This means feature locality is to increase consequently due to accuracy improvement due to noise injection.

3.3.3 Visualizing features

Visualizing the predicted class scores for network inputs highlights the discriminative object parts detected by the CNN which gives reasons for accurate or inaccurate classification. We use feature maps and class activation maps to localize deep features that offer understanding of discrimination used by CNNs. We Visualized each layer results in feature vectors that come with different characteristics. We believe that noise encourages the network to identify the extent of the object as compared to the noiseless case which encourages it to identify just one generalization of the target. The learning rate is suppressed when the output has uncertain value and is accelerated when it has a definite value. We verify this experimentally on KDEF and CK+ datasets in experiment section. While noisy training achieves slightly outperforms classification performance of noiseless case, noisy case significantly outperforms noiseless for localization.

It's worth mentioning that the class activation visualizes certain class at a time. Let's say we are visualizing emotional state E , visualizing the activations of the units that leads to classifier E shows significantly higher values while other class activations converge to zero. For illustration, in Figure 3.2 we visualize activation maps for neutral and anger. Class activation map localizes the informative regions for those classes. Activations with low values (> 0.015) are ignored during the visualizing as suggested on Zhou et al. [129]. The network is focusing on the eyes when giving the decision about neutrality while it looks both at eyes and mouth (specifically teeth presence) when classifying anger. More schemes are discussed in the experiments section.

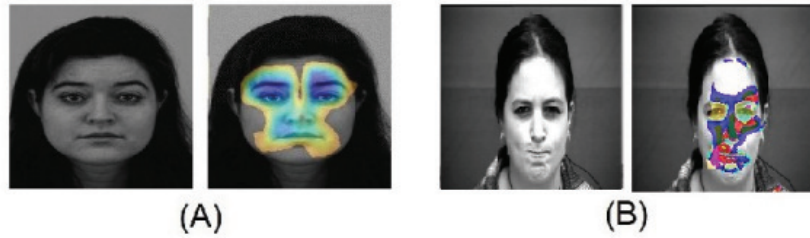


Fig. 3.2 This Figure shows the different emotions class activation maps of KDEF and CK+ datasets. The emotions are given in the following order: A-Neutral from KDEF and B-Anger from CK+. The blue region indicates higher activation which means more attention is given by the network to make classification decision.

We visualize feature maps of the weights that connect the last layer classifier with the convolutional units ($MaxPool3_B$) as shown in figure 3.3. In convolutional networks, a window with the filter size slides across the input searching for salience features. That way we can find features in that window. Feature maps gradually capture higher level features that consist of lower level features. Convolution units result in many filter outputs. In every case, we visualize unit outputs with the highest activations. Those units are the units to make the decision later by supplying those high activations to the classifiers. Figure 3.3 shows feature maps for some emotion states without noise injection for illustration. In figure 3.3-A, the network observes the lips and eye shapes to detect fear. Higher weights in those regions indicate their importance. In figure 3.3-B, for instance, the eyebrows have higher weights than any other region which gives an indication of its importance when classifying anger. More schemes are considered and discussed in the experiments section.

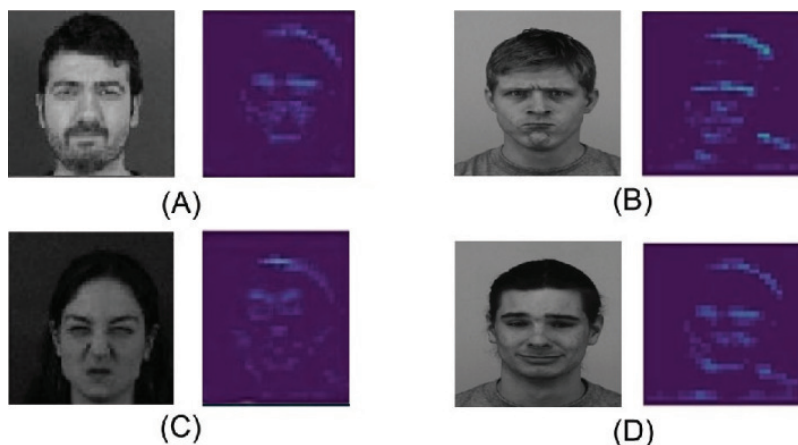


Fig. 3.3 This Figure shows the different emotions feature maps for samples of KDEF and CK+ datasets. The emotions are given in the following order: A-Surprise(CK+), B-Anger(KDEF), C-Fear(CK+) and D-Neutral(KDEF).

3.4 Experiments

In our experiment, we train and evaluate classification network using KDEF[46] and CK+[141] datasets. We evaluate the effect of noise injection during training. Beside accuracy as an important measure of evaluation, we analyze feature localization. Extracted Features for happiness are different from features for sadness. We propose two classification schemes to clearly visualize the noise effect on feature localization; A Single emotion expert and multiple emotions expert. In the single emotion expert, the classifier size is 2 (emotion E and others). In the multiple emotions expert case, the classifier size is 7 to include all emotional states. The single emotion expert classifies a single emotion state and all other states (e.g. Happiness and all others). Our classification accuracies for KDEF without noise vary between 93.0% to 98.95% and losses of 0.0012 to 0.0009 depending on the emotional state. The multiple emotions expert detection rate was 89.73% while the loss was 0.013 without noise. Classification accuracy For CK+ single emotion expert without noise varies between 93.25% to 99.08% and losses of 0.0005 to 0.0002 depending on the emotional state. The multiple emotions expert detection rate for CK+ was 92.65% while the loss was 0.0008 without noise. Experiments details and results are shown below. Analysis of the findings is found in the discussion section.

3.4.1 Dataset

We used KDEF and CK+ datasets in our experiments. The KDEF dataset has a population of 35 females and 35 males between 20 and 30 years of age. No beards, mustaches, earrings or eyeglasses, and no visible makeup. Subjects evoked the emotion that was to be expressed while seated three meters from the camera. The dataset comes with different angles describing 7 emotional states.

The CK+ facial expression database includes 486 sequences from 97 posers. Every sequence begins with a neutral expression and end up with one of the 6 expressions. The apex and offset of sequences are used. For both datasets, the considered emotions are happiness, surprise, neutral, fear, sadness, disgust and anger. We specifically analyzed the improvement in localization with noise injection for portrait pictures. In our experiment we used 1960 portrait images from KDEF dataset and 900 portrait images for CK+ dataset. Figure 3.4 and 3.5 show sample data.

KDEF Dataset comes with a variety of the expressions depth, making it one of the challenging classification subjects [46]. The idiosyncratic hit rate was given by authors. It is defined by the percentage of correct emotion prediction by an observer. The mean success index was 71.87%, ranging over the different emotions from 43.03% to 92.65% [46].



Fig. 3.4 This Figure shows the different emotions given by KDEF. The seven emotional states are shown in random order. The states are happiness, surprise, fear, sadness, disgust, anger and neutral.

3.4.2 Single emotion expert case

Our network receives Images with 224×224 pixels which defines the input size of the network. We consider the network accuracy and overall loss beside the localization ability for every case. The location of the feature and the high activations around important regions are visualized. We also evaluate the training loss and testing accuracy of our model. The training took 100 Epochs with a mini-batch size of 30. The loss function for noisy and noiseless training is the SoftMax cross entropy.

In this experiment, we used 130 images of class E of KDEF dataset and 800 images of other classes. We used 1000 images for testing. We used SGD optimizer with learning rate 0.0001 without weight decays. We inject the Gaussian noise into the Inception model blocks. The noise we used comes with a mean of 0 and the standard deviation $\sigma = 0.1$ based on multiple experiments. The noise was injected during training and evaluated during testing through the accuracy of classification, loss performance and localization abilities. We conducted two experiments by varying the injection point. The first was conducted by adding noise between $Conv2_A$ and $conv2_B$. The second was conducted while adding noise between $Conv3_A$ and $conv3_B$.

Classification results for noiseless scheme beside different noise injection positions are shown in table 1. We refer to our stacked Inception classifier as SIC. The cascade use of



Fig. 3.5 CK+ emotional dataset contains mixture of colored and gray-scaled emotional transitions.

neural networks for facial expression recognition given by Kukla [58] is referred to as CAS in this chapter. We refer to the Deep learning approach given Alessandro [57] as DLE. Finally, we refer to the model given by Garcia et. al [48] as SVM. CAS, SVM and DLE works are considered the state of the art when considering KDEF dataset for facial expression analysis.

The network benefits from injecting noise to the first block more than the second block. The margin of improvement between the two schemes is minor with a value of $\pm 0.2\%$. We discuss this phenomenon in the next section. SVM model shows the highest accuracies in most of the cases. Compared to SVM model, our model shows higher accuracies for anger VS others with 97.2% and 97.1% for the first block and second block noise injections respectively. It also shows better accuracies for surprise vs others with 96.5% and 96.32% for the first block and second block noise injections respectively. Within our model, the lowest case is for fear with 93.04% and 92.8% for first block and second block noise injections respectively. It's worth mentioning that dual injection for both joints at the same time didn't improve the accuracy more that injecting noise at the first block scheme.

For the CK+ experiment, we used 100 images of class *E* and 500 images of other classes. We used 300 images for testing. Same KDEF experimental setup is used. Classification results for noiseless scheme beside different noise injection positions for CK+ are shown in table 2. Similar to KDEF findings, the network benefits from injecting noise to the first block the most. Best accuracies for neutral vs others with 99.52% and 99.10% for the first block and second block noise injections respectively. The lowest accuracy is for fear with 95.11%

Table 1 The results of our experiments on KDEF dataset for Emotion Vs. all other emotion states is shown beside Alessandro et al.[57] work denoted by DLE, Kukla et. al[58] work denoted as CAS and Garcia et. al [48] work denoted as SVM. Each classification task is done independently. Happiness vs Others is abbreviated as HA, Surprise vs Others is abbreviated as SU, Neutral vs Others is abbreviated as NE, Fear vs Others is abbreviated as FE, Sadness vs Others is abbreviated as SA, Disgust vs Others is abbreviated as DI, Anger vs Others is abbreviated as AN.

E	SIC			DLE	CAS	SVM
	No Noise	1 st ICP Noise	2 nd ICP Noise			
HA	97.8%	98.1%	97.9%	86.3%	80.83%	100%
SU	96.3%	96.5%	96.32%	92.5%	78.33%	95.24%
NE	98.95%	98.90%	98.99%	81.5%	80.12%	100%
FE	93.0%	93.04%	92.8%	81.03%	82.5%	97.80%
SA	95.2%	96.21%	95.0%	91.3%	78.33%	96.47%
DI	94.6%	95.16%	94.69%	88.7%	70.0%	98.97%
AN	96.8%	97.2%	97.1%	85.6%	88.33%	91.36%

and 94.24% for first block and second block noise injections respectively. This shows that the advantage of injecting noise is independent from characteristics of the dataset.

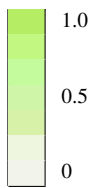
In figure 3.6 and 3.7, we visualize the feature maps and the class activation for noiseless and multiple noisy trainings. We observe stronger feature representation when noise is injected into network whether the injection is at first or second Inception block. Figure 3.6 shows how the internal representation of the features improves due to noise injection for both KDEF and CK+. 3.6-A, for instance, is evaluating fear. The shape of the lips is dominating the decision-making process with significant higher weights upon injecting noise. 3.6-C, on the other hand, is analyzing anger. After injecting noise the eyebrows region gets higher weights. In all cases, the third column shows best representations.

On the same hand, the localization abilities of the network significantly become restricted towards more critical regions after injecting noise. This is observed via class activation mapping in figure 3.7 for both KDEF and CK+. Considering surprise emotion for instance in 3.7-A, the network observed almost the entire face to give its prediction. After injecting noise, the region has got smaller. This is a significant improvement considering the accuracy has improved as well. Each emotional state benefits from the noise based on its local characteristics. Anger classification focuses on the teeth and eyebrows. Disgust classification focuses on the mouth and eyes opening. sadness classification focuses slightly more on the lips shape and so on. Once again, the noise injection at the first Inception block (Third column in figure 3.7) benefits feature localization for all emotions.

Table 2 The results of our experiments on CK+ dataset for Emotion Vs. all other emotion states is shown. Happiness vs Others is abbreviated as HA-Others, Surprise vs Others is abbreviated SU-Others, Neutral vs Others is abbreviated as NE-Others, Fear vs Others is abbreviated as FE-Others, Sadness vs Others is abbreviated as SA-Others, Disgust vs Others is abbreviated as DI-Others, Anger vs Others is abbreviated as AN-Others.

Emotion	No Noise	1 st block noise	2 nd block noise
HA-Others	98.02%	98.63%	98.22%
SU-Others	97.01%	97.86%	96.63%
NE-Others	99.08%	99.52%	99.10%
FE-Others	93.25%	95.11%	94.24%
SA-Others	96.01%	96.82%	96.21%
DI-Others	95.04%	95.16%	94.11%
AN-Others	97.82%	97.91%	97.85%

Table 3 The confusion matrix on the KDEF Dataset is shown without injecting noise into the network. The lowest accuracy is achieved by the emotion fear with 83% while happiness is detected with 95.4%. Happiness is abbreviated as HA, Surprise as SU, Neutral as NE, Fear as FE, Sadness as SA, Disgust as DI and Anger as AN.

	HA	SU	NE	FE	SA	DI	AN	
HA	0.954	0.015	0.01	0.0	0.006	0.0	0.015	
SU	0.013	0.951	0.00094	0.011	0.016	0.0006	0.00746	
NE	0.011	0.034	0.914	0.00311	0.0009	0.014	0.02299	
FE	0.008	0.015	0.0295	0.83	0.025	0.0551	0.0374	
SA	0.00059	0.0405	0.022	0.03	0.8799	0.0025	0.02451	
DI	0.023	0.015	0.021	0.039	0.032	0.831	0.039	
AN	0.021	0.023	0.016	0.016	0.001	0.002	0.921	

3.4.3 Multiple emotions expert case

In this experiment, we used 130 images per class for training and 1000 images for testing from KDEF dataset. We utilized SGD optimizer with learning rate 0.0001 without weight decays. The proposed architecture has proven to be very effective on this dataset with an average accuracy of 90.6%. We inject the Gaussian noise into the Inception model blocks. Similar to the previous experiment, we varied the injection points. The first was conducted by adding noise between *Conv2_A* and *conv2_B*. The second was conducted while adding noise between *Conv3_A* and *conv3_B*. The confusion matrix in the table 3 shows results for the noiseless training case. Confusion matrix at table 4 shows results of the first Inception block noise injection. Table 5 shows results for injecting noise at the second Inception block.

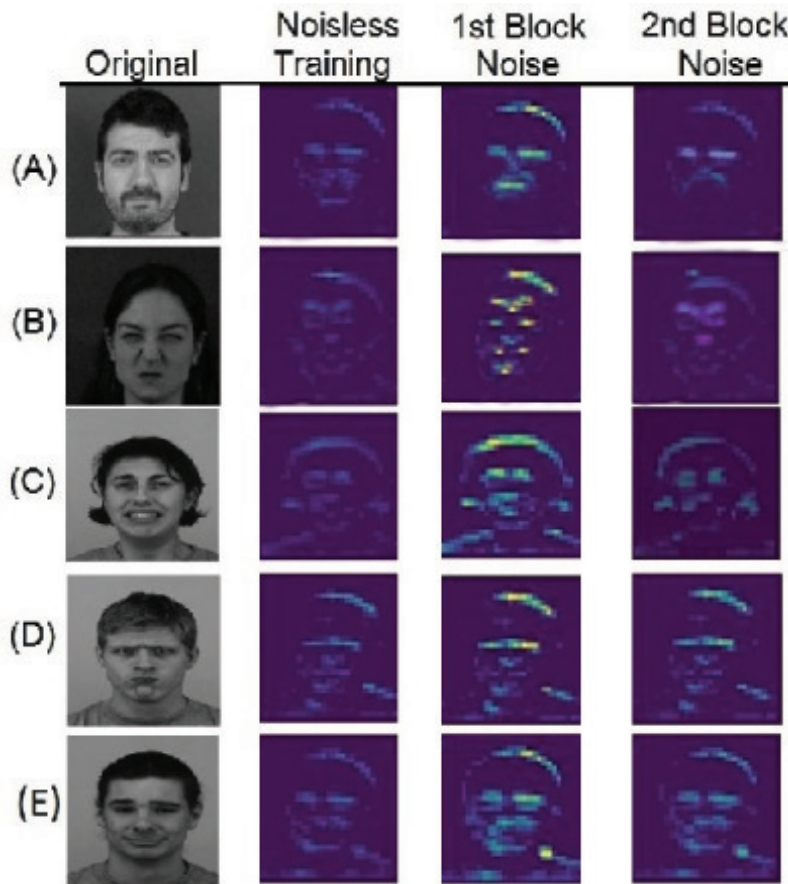


Fig. 3.6 This Figure shows the different emotions feature maps of KDEF and CK+ datasets. The emotions are given in the following order: A-Contempt (CK+), B-Disgust (CK+), C-Fear(KDEF), D-Anger(KDEF) AND D-Neutral(KDEF).Saliency is key when determining the efficiency of a feature map representation over the other. First block noise column is showing best feature representations.

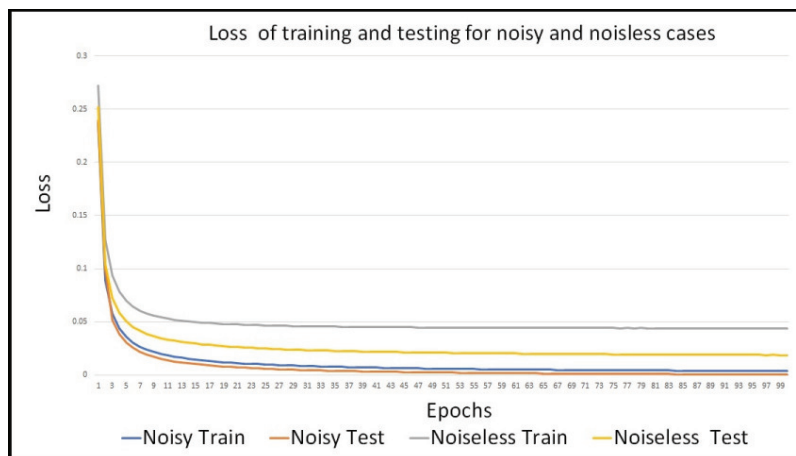
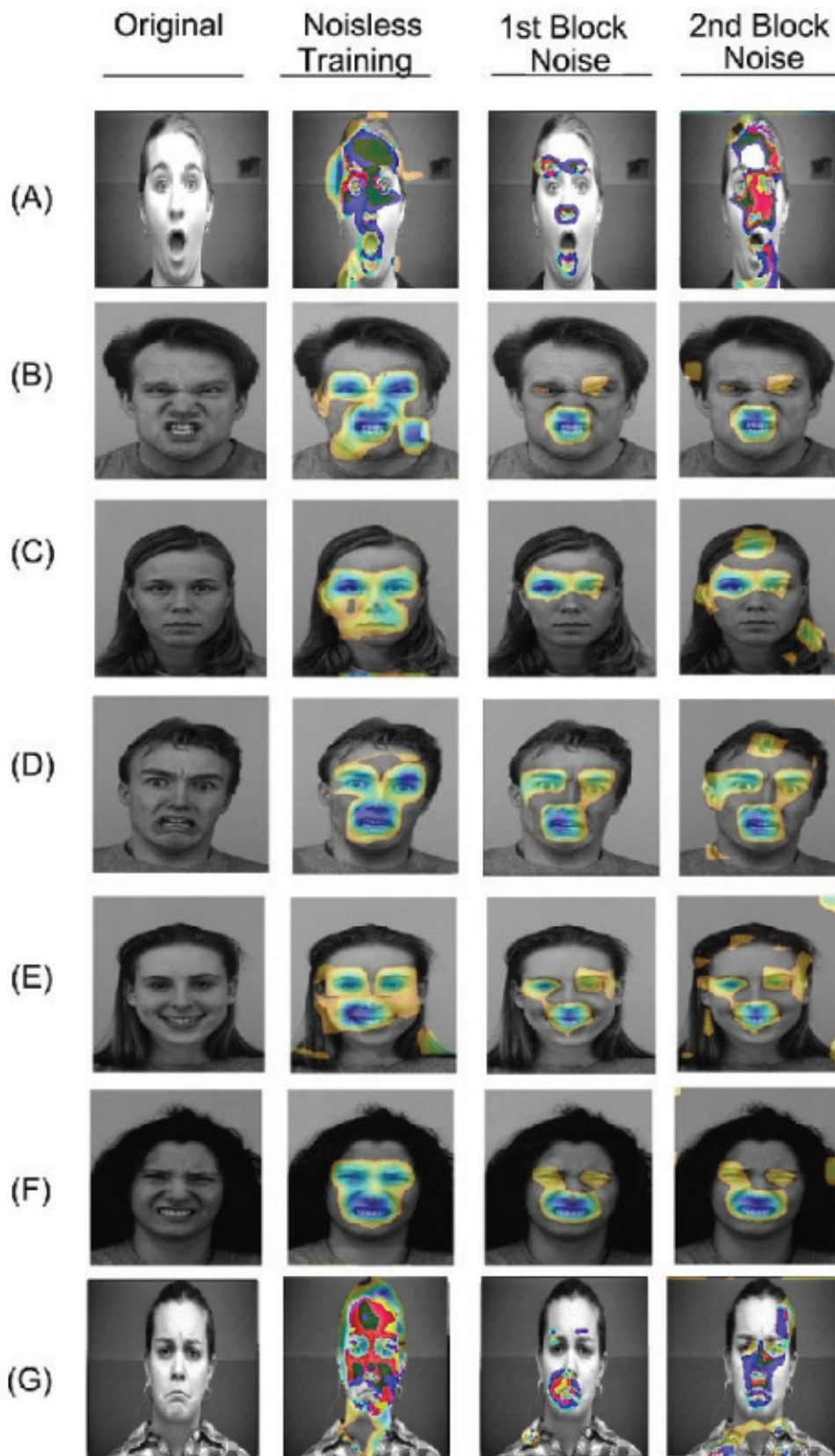


Fig. 3.8 Chart shows the network loss during KDEF training and testing for the noisy and noiseless training. It's observable that the loss is having less value much faster in the noisy case. The varying between training and testing curves is natural due to the learning tendencies for the SGD optimizer.



b

Fig. 3.7 This Figure shows the different emotions class activation maps of KDEF and CK+ datasets. The emotions are given in the following order: A-Surprise(CK+), B-Anger(KDEF), C-Neutral(KDEF), D-Fear(KDEF), E-Happiness(KDEF), F-Disgust(KDEF) and G-sadness(CK+). Features locality gives an indication which regions the network depends on for decision making

Table 4 The confusion matrix on the KDEF Dataset is shown while injecting noise into the first Inception model during training. The lowest accuracy is achieved by the emotion fear with 88.7% while happiness is detected with 96.7%.

	HA	SU	NE	FE	SA	DI	AN
HA	0.965	0.015	0.02	0.0	0.0	0.0	0.0
SU	0.01	0.967	0.00094	0.009	0.005	0.0006	0.00746
NE	0.009	0.03	0.942	0.002	0.011	0.006	0.0
FE	0.007	0.01	0.02	0.8989	0.015	0.04	0.0091
SA	0.001	0.04	0.021	0.03	0.9035	0.001	0.0035
DI	0.019	0.012	0.001	0.032	0.012	0.887	0.037
AN	0.02	0.02	0.012	0.015	0.002	0.001	0.93

Table 5 The confusion matrix on the KDEF Dataset is shown upon injecting noise into the second Inception model. The lowest accuracy is achieved by the emotion fear with 84.11% while happiness is detected with 96.0%.

	HA	SU	NE	FE	SA	DI	AN
Ha	0.96	0.017	0.023	0.0	0.0	0.0	0.0
Su	0.01	0.956	0.0104	0.02	0.003	0.0006	0.0
Ne	0.015	0.03	0.925	0.005	0.02	0.005	0.0
Fe	0.015	0.03	0.03	0.8411	0.02	0.04	0.0239
Sa	0.011	0.049	0.021	0.031	0.8835	0.001	0.0035
Di	0.0592	0.02	0.002	0.035	0.014	0.8528	0.017
An	0.020	0.0214	0.017	0.015	0.002	0.001	0.9236

We used the same experimental setup for CK+ dataset. We used 100 images per class for training and 200 images for testing. The experiment resulted in average accuracy of 90.6%. We inject the Gaussian noise into the Inception model blocks. Similar to the previous experiment, we varied the injection points. The confusion matrix in the table 6 shows results for the noiseless training case. Confusion matrices at tables 7 and 8 show results for injecting noise at first and second Inception blocks respectively.

Finally and to confirm the improvement in accuracy and loss after injecting noise. Figure 4.2 shows the training accuracy in both noisy and noiseless cases while figure 3.8 shows the loss in both cases for 100 epochs training for KDEF dataset. Figure 3.11 and 3.10 show the accuracy and loss for both noisy and noiseless case for CK+ dataset.

Table 6 The confusion matrix on the CK+ Dataset is shown without injecting news into the network. The average accuracy is 92.65%.

	HA	SU	NE	FE	SA	DI	AN
HA	0.963	0.005	0.02	0.0	0.0	0.01	0.002
SU	0.01	0.9611	0.00094	0.009	0.005	0.0006	0.00746
NE	0.009	0.03	0.942	0.002	0.011	0.006	0.0
FE	0.007	0.01	0.02	0.8989	0.015	0.04	0.0091
SA	0.001	0.04	0.021	0.03	0.9035	0.001	0.0035
DI	0.019	0.012	0.001	0.032	0.012	0.887	0.037
AN	0.02	0.02	0.012	0.015	0.002	0.001	0.93

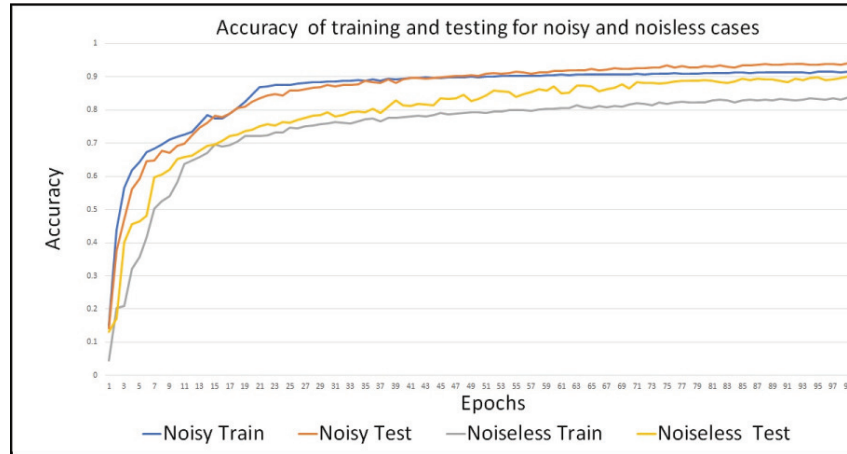


Fig. 3.9 KDEF Accuracy chart shows the improvement of the network learning through epochs. With the two cases, we trained 100 epochs with and without using noise. The noisy case shows observable improvement over the noiseless case.

Table 7 The confusion matrix on the CK+ Dataset is shown while injecting noise into the first Inception model during training. The average accuracy is 94.34%.

	HA	SU	NE	FE	SA	DI	AN	
HA	0.982	0.01	0.008	0.0	0.0	0.0	0.0	1.0 0.5 0
SU	0.001	0.971	0.0184	0.0054	0.0011	0.0001	0.003	
NE	0.0013	0.03	0.961	0.003	0.003	0.002	0.0	
FE	0.011	0.03	0.03	0.9021	0.0048	0.021	0.0011	
SA	0.0032	0.004	0.017	0.037	0.937	0.0001	0.002	
DI	0.0069	0.04	0.005	0.017	0.003	0.911	0.017	
AN	0.002	0.034	0.007	0.014	0.002	0.0001	0.94	

3.5 Discussion

The accuracy of classification on the KDEF and CK+ datasets shows that the proposed architecture is robust and comparable to the state of the art methods. Our method shows better performance against Alessandro et. al [57] and Kukla et. al [58] approaches. Compared to our work, the SVM model proposed by Garcia et. al [48] results in better detection accuracies for 5 out of 7 emotion classification. It's worth mentioning that the preprocessing phase of their work is unavoidable. The face detection and Gabor filtering are expensive tasks considering real-time processing. The training time in our work is time consuming. The produced model processes videos frames in a real time manner offering a comparable accuracy.

Misclassification usually occurs because the network is looking at undesirable regions of the image. Some undesirable regions won't necessarily cause misclassification but they definitely affect the accuracy of the neural network on the long run. Initially, without any interference with the training, the network starts observing regions within subject's faces that

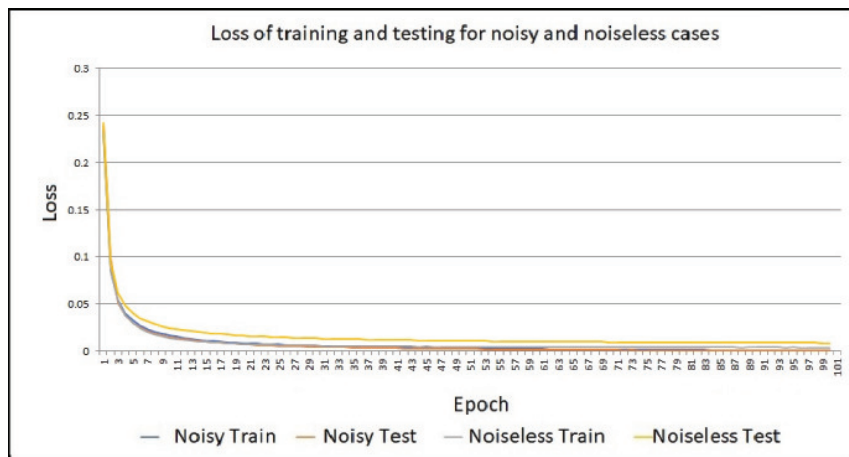


Fig. 3.10 Classification loss is shown during training and testing of CK+ for the noisy and noiseless training.

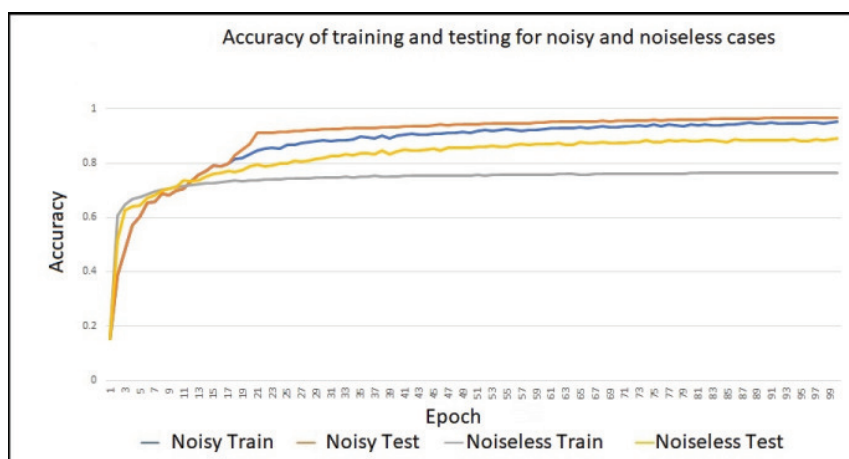
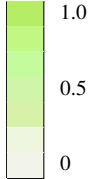


Fig. 3.11 The classification accuracy of CK+ shows the improvement through epochs.

Table 8 The confusion matrix on the CK+ Dataset is shown upon injecting noise into the second Inception model. The average accuracy is 93.59%.

	HA	SU	NE	FE	SA	DI	AN
HA	0.977	0.01	0.008	0.0	0.0	0.0	0.005
SU	0.0005	0.9623	0.0184	0.0094	0.0012	0.0051	0.0031
NE	0.006	0.0239	0.9511	0.007	0.004	0.0026	0.0
FE	0.031	0.023	0.003	0.9	0.0049	0.026	0.0121
SA	0.0038	0.0	0.016	0.0573	0.921	0.0001	0.0018
DI	0.007	0.04	0.005	0.017	0.013	0.901	0.017
AN	0.002	0.0344	0.007	0.0144	0.0021	0.0011	0.939



are critical even for humans to give decisions about an emotional state. Those regions, as we illustrated in figure 3.2, are the eye region, nose and mouth. Later after injecting noise, those regions become more restricted to more important areas. Regions like eye corners, laugh lines beside the nose or marionette lines below the lips get higher importance. The darker the color of the activation map, the higher the activation. Our network detected the importance of those regions without any further training on where to look.

To determine the best location for noise to be injected into our model, we consider accuracy as a measure. From tables 4 and 7, accuracy is the highest when noise is injected in the first block offering special features for the training that was given to the second block. To understand why the second injection scheme isn't as efficient as the first injection scheme we can observe the network visualization. Many of the activation maps are showing disturbance on their outputs that was delivered to the classifier which results in less accuracy. We think this is because the network didn't benefit long enough from the noise and was directly tunneled to the classifier which has many filter outputs to be processed. This resulted in minor improvement in accuracy and localization but wasn't as powerful as the previous injection scheme. This also clarifies why injecting noise into both (first and second block) points does not benefit the network performance to be better. The feature localization doesn't only tell us where the network is looking to make a decision but also how important is that region in making this decision. In figure 3.7, facial expressions of anger, fear and surprise reside in the same region which is mainly around the mouth. Many researches indicated that those 3 emotional states tend to be confused with each other by machine learning algorithms because of their close locality. Our activation maps are showing unique visualization for every state which justifies why accuracies in confusion matrix in tables 4,5,7 and 8 are better than confusion matrix in tables 3 and 6.

The network shapes its representation gradually to give an approximation of the input. The feature map visualization shows a weak representation of the input without noise. The results are good in such case but once the noise is injected the features become more salient. Injecting noise at the first block benefits the network into building up a structured

representation which is used to train the second block giving an accurate feature map as shown in the first block noise column in figure 3.6. When the noise is injected into the second block, the network uses it to improve the already existed representation and didn't benefit much as shown in figure 3.6. Feature maps show predictable behavior as mentioned in [43] when noise is injected. It achieves the ability to generalize the network decisions through automatic structuration.

3.6 Conclusion

In this chapter, we investigated the effect of injecting noise on enhancing localization of facial features and improving generalization of CNN made of GoogleNet Inception blocks and a classifier. We proposed detailed explanation through feature map and class activation maps representation on why injecting noise at certain locations increases the network performance. Experimentally, injecting noise at those locations offers better performance for both deep and shallow networks. Trained CNNs has learned to perform object localization and highlighting the discriminative object parts, without using any bounding box annotations.

Next chapter extends the extraction of local features to sequential relationship in the temporal domain that appears due to the natural onset apex offset during an emotional transition. Besides modeling analysis, we discuss the importance of how data feeding is important to the learning and understanding of emotions.

Chapter 4

Emotions intensity estimation by Siamese and Triplet networks

4.1 Background

Every day we perform hundreds of nonverbal behaviors to exchange information through body language, facial expressions and paralanguages. Each behavior is delivered in parallel with an emotion paired to it ([159]). Understanding those emotions deepens our communication and enriches our discussions. Such emotional awareness is important for computers as well to allow more natural human-machines interaction ([168]).

Many researches have tackled the recognition of emotions in videos and pictures ([148, 161, 130]). Subject facial expressions are used to predict an emotional state. Another trending emotional analysis area is facial expression intensity estimation ([166, 170]). It helps defining how much a person is influenced by experiences such as discomfort in a medical treatment or joy while watching an advertisement.

The diversity of patterns in facial expressions beside the absence of standard way to label intensities make intensity estimation a challenging task ([143]). Many researches propose methods based on Facial Action Coding System proposed by [151] (FACS) which describes the facial expression using facial actions such as contracting glabella or squinting outer canthus. Such researches aim to classify an expression in an image or video as one of the six basic expressions ([144, 134]). Regardless of those methods good performance, they neglect the dynamics of an expression which is critical in interpreting it's meaning ([167, 150, 128]). In this chapter, we propose a Convolutional Neural Networks (CNN) based method to interpret the dynamics of a facial expression through it's intensity in the temporal domain.

CNN's have been successfully used to achieve state-of-the-art performance in a variety of pattern recognition tasks such as object classifications, feature localization, image regeneration, and speech recognition. However, with weak supervision on the training, predictions

given by CNN tend to break down. [140] et al. proposed generalization method to recognize these unfamiliar categories through Siamese networks. This network structure has improved the network learning capabilities by exploiting additional information about how the training samples are related. With their abilities to minimize the distance, those networks were used in many fields such as signature matching ([136]), Image originality verification and media search retrieval ([142]).

However, the representations learned by these models provide below average results when used as features to learn explicit representations. Triplet network model learns representations that are comparable with a network that was trained explicitly to classify samples ([135]). This model doesn't require knowing the class of the processed input, it needs to know that only two out of three inputs are sampled from the same class, rather than knowing what that class is. This model learns using only comparative measures instead of labels ([165]).

Inspired by Siamese and triplet network abilities, we analyze facial expression intensity estimation against the basic emotion states of anger, happiness, sadness, surprise, contempt and disgust. We also analyze the evolution of the emotion in terms of the emotional features locality ([129]) and detect Micro-Expressions. We generate activation maps and feature maps ([145]) to visualize the internal representation of the network. To the best of our knowledge Siamese and triplet networks abilities haven't been used in emotion intensity estimation. Our model receives sequences that contain emotional transitions and estimates their intensity level. Our experiments steadily result in accuracy improvements and clearer localization abilities of important features. For instance, a gradual increase in activation around happiness facial features such as mouth and cheeks or stressing facial features such as glabella. Our training of the neural network aims to estimate emotional intensity of the 7-emotional states in the [141], MUG dataset ([149]) and MMI dataset ([146]) and detect Micro-Expressions in CASME, CASME II and CAS(ME)² datasets([163],[164] and [153]). We ordered the data using apex to onset to apex format ([166]). Our intensity estimation performance excels in terms accuracy, generalization for the hidden unit's activation and feature localization when compared to other intensity estimators. This can be observed through experiments and visualizations for the hidden layer representation. Those findings will help understanding emotional transitions by looking at the evolution the action units that define an emotion.

This chapter is organized as follows. In section 2, relevant models of Siamese and triplet networks are reviewed as related work. The details of the approach used in this chapter are explained in section 3. The model definition for Siamese and triplet training is mentioned in section 3.2. Experiments details are described in section 4. Discussion of the results is in section 5. Finally, the conclusion is shown in section 6.

4.2 Related work

Expressions intensity estimation issue is gradually gaining research attention. This attention comes with more awareness of the challenges that are paired with this issue, specifically the absence of leveled or annotated intensity data model ([170, 166, 159, 133, 157, 98]).

4.2.1 Expressions Intensity Estimation

Some studies analyze the facial landmarks and their relationship with certain expression intensity. [139] applied SVM and a cascade neural network to classify facial landmarks. [137] reduced feature dimensions using kernel principle component and local linear embedding and then trained those features by SVM. Both approaches result in competitive loss and performance but fail to generalize for the unseen data and analyzed intensity in a descriptive rather than quantitative way.

In another attempt to overcome unlevelled data, [170] considered intensity estimation as a regression problem for frame-level. They used ordinal relationship across an emotional transition sequence. Their work is found to be valuable when the intensity annotation is not available for all the frames. [166] used the RankBoost for facial expression recognition and intensity estimation considering it as ranking problem. Intensity level is scored by the ranking function. They both consider an emotion transition as a sequence that goes from the onset, to apex, then onset again. This highlights the fact that the expression intensity has monotonous increase yet the details vary from subject to another.

Both [170, 166] result in the state of the art estimation. Considering the large number of action units (AU's) to be analyzed, there is still space to generalize the intensity estimation. In this chapter, we aim to handle intensity estimation through neural networks which is known as a general approximators. Our work is based on the ranking model which extracts the ordinal relationships between sequenced data. We utilize both Siamese and triplet networks to highlight all changes that refer to intensity change.

4.2.2 Siamese Network

Networks with multiple parallel paths have been utilized to find similarities or differences between inputs ([152, 158]). Siamese network ([136]) is a special architecture consists of two sub-networks that share parameters. One property of Siamese neural network is it's unified and clear objective function ([131]) which allows learning optimal metric towards the target automatically. Many good experimental results have been obtained in the field of verification and person identification with this model ([152, 158, 136]).

[169] have compared Siamese MLP network with the classical MLP for face identification, showing that the Siamese MLP training with side information achieves comparable classification performance with the classical MLP training on fully labeled data. [158] experimented Siamese network for verification in which the number of classes is very large. They proposed a discriminative loss function that drives the system to make the right decision.

On the other hand, Siamese network models perform weakly when learning explicit representations ([135]). Triplet networks solve this issue and show ability to learn representations that are comparable to classification networks ([165]).

4.2.3 Triplet Network

Current Siamese and triplet networks compute the gradient from individual pairs or triplets of local paired image patches. Pairwise ranking model is a widely used learning-to-rank formulation. Generating good triplet samples is a crucial aspect of learning pairwise ranking model. [162] showed that a combination of the triplet and global losses produces significant embedding enhancements. They used global loss that minimizes the overall classification error in their training set, which can improve the generalization capability of the triplet networks. [138] employs a triplet-based hinge loss ranking function to characterize fine-grained image similarity relationships.

Considering the natural monotonic flow from onset to apex in every emotional state, we assume with the proper loss function, Siamese and triplet network will give robust intensity estimation and will be able to generalize unseen data.

4.2.4 Micro-Expressions

Micro-expressions are brief spontaneous facial expressions triggered when a person encounters an emotional transition. Micro-expressions are visually similar to facial expressions but are short and repressed which makes detecting them a challenging task. Detecting Micro-expression is important for emotional classification and many approaches has been used for the detection process. [154] et al. detected micro expressions using Gabor filters and used Support Vector Machine (SVM) to recognize them. [160] et al. developed a 3D gradient descriptor and used k-means algorithm to classify onset, apex and offset frames. In our work, considering the Siamese and triplet networks abilities to detect minor changes between consecutive frames, we evaluate their ability to detect Micro-expressions in the well-annotated datasets of CASME, CASME II and CAS(ME)² ([163],[164] and [153]).

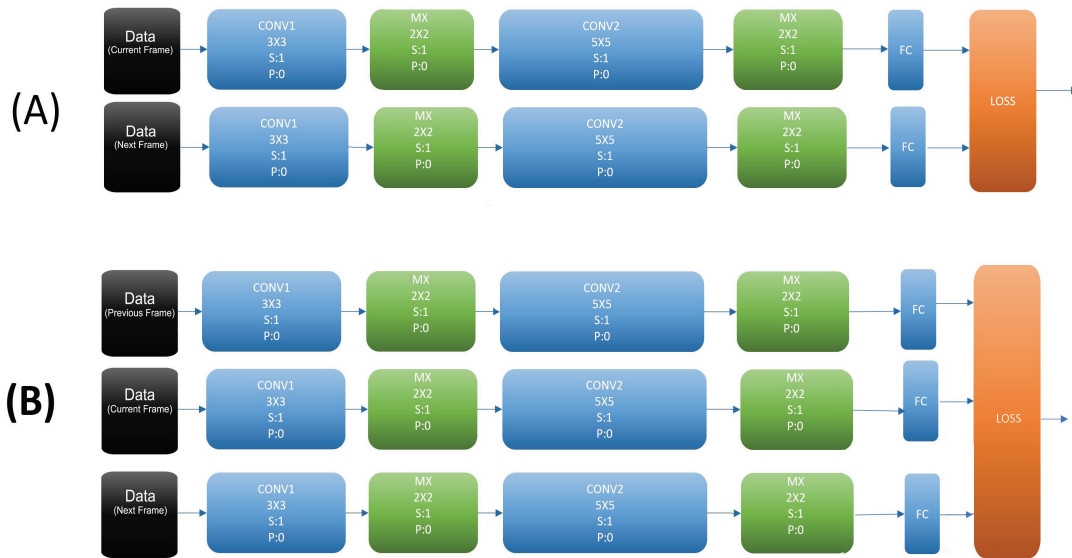


Fig. 4.1 The proposed architecture shows the Siamese and triplet models connected to the exponential loss, the kernel size is shown on every block and S represents the Stride and P represents the Padding.

4.3 Proposed method

In this chapter, we evaluate the Siamese and triplet convolutional neural networks abilities as ranking models of facial expressions intensity. In ranking problem, it's typically assumed that we have a space in which the data sequence exists with ordered ranks ([156, 166]). Naturally the ranking can be represented in a continuous function which describes the relation between certain input and it's location in a sequence. Considering most of the available datasets don't give quantitative intensity value, ordinal relationship between pairwise data is found useful. The pairs can be organized in the ordinal pairwise format according to the temporal variation of expression. The proposed models are summarized in figure 4.1. Figure 4.1-A shows the Siamese network which receives pairwise data stream. Figure 4.1-B illustrates the triplet network which is fed with triplet-paired data. We will discuss how to generate the paired data and how the distance is minimized between the pairs through exponential loss function shortly.

4.3.1 Data Setup

The ordinal relationship between pairs can be easily labeled considering their temporal order. [166] used an efficient ordering and pairing technique that emphasizes the smoothness and ordinality of an emotional transition. We used three datasets in our experiment: [141]



Fig. 4.2 The organization of training data on the ranking of intensities of all emotional transitions for CK, MUG and MMI datasets. The sequence on top shows how Siamese pairs are created while the sequence at the bottom shows how triplet items are created. Every sequence starts from an emotional peak and ends at another emotional peak passing through neutral state.

(CK), Multimedia Understanding Group (MUG) given by [149] and MMI Facial Expression Database proposed by [146]. A robust strategy must be considered to generate longer sequences with the smoothest transitions between consecutive pairs. The transition must generate pairs and triplets whether its leading to an apex or onset. Given that every emotion has an apex and onset, two expressions can be connected for the same person forming an apex to onset to apex transition. We used data organizing strategy given in [166] in which expression E_i with it's sequence Q_j starts from apex to onset is followed with the reversed sequence of expression E_j . This forms an extended sequence that goes from apex to apex for the same person. From this sequence we built the pairwise instances $\{x_0, x_1\}$ for the ranking model. In case of Siamese network data, two consecutive pairs are taken at a time. For triplet network data, three items are taken at a time $\{x_0, x_1, x_2\}$ described as negative, anchor and positive respectively. Figure 4.2 shows an example of the pairs generation for both Siamese and triplet networks.

4.3.2 Siamese and triplet networks

Facial expressions are gradually changing while expressing an emotion. Each AU appears at certain level during this process. The dissimilarities between consecutive pairs during an emotional transition highlight those AU's and therefore give an estimate of expression intensity. Siamese network consists of two parallel paths that are efficient in finding how pairs are similar. In our model, pairs and triplets are gray scaled beforehand. Each input is fed to filters of the first Convolution which analyzes the local regions of the input image and groups into filter banks. This results in numerous clusters with different activation levels. The filter's

output is then down-sampled by the max pooling. The same happens again with different filter sizes to view every image with different window size. Each branch within the Siamese and triplet networks sees it's input through same filter sizes which enables the network to detect minor changes. The detected salient features are forwarded to fully connected layer and later loss function is used to minimize the distance between the two pairs or triplets.

The process of learning differences between pair items is based on the Rankboosting used by [132, 166]. The Rankboost incrementally chooses weak rankers by minimizing the exponential loss function. This is found useful in our case considering the nature of AU evolution during an emotional transition. Therefore, The loss function we used is the exponential loss, lets refer to each branch output as $g_i(x_i)$ where x_i is pair instance item and i is the index of the branch, therefore the loss of the Siamese network is given in (1) while the triplet network loss is given in (2):

$$Loss_{Siamese} = \min \sum_{x_0, x_1} \exp(g_0(x_0) - g_1(x_1)) \quad (4.1)$$

$$Loss_{Triplet} = \min \sum_{x_0, x_1, x_2} \exp(g_0(x_0) - g_1(x_1)) - \exp(g_0(x_0) - g_2(x_2)) \quad (4.2)$$

During the Siamese and triplet network training, each image within a pair enters a 3×3 kernel convolution with a stride of 1 and padding of 0 followed by a 2×2 max pooling with stride of 1 and padding of 0 to down-sample the different filters resulted from the previous convolution. The down-sampled data then go through 5×5 kernel convolutions with a stride of 1 and padding of 0 followed by a 2×2 max pooling with stride of 1 and padding of 0. The order of the 3×3 and 5×5 kernels allows salient features to be captured regardless of their size. The output is then tunneled to a fully connected layer that forwards the data to the exponential loss.

4.3.3 Visualizing features

It is interesting to visualize the discriminative regions that the network uses to find the differences to estimate an emotion intensity. It helps determining whether the network is looking at the correct location while estimating the expression intensity as well. We use feature maps similar to [145] and class activation maps illustrated by [129] to localize deep features that offer understanding of discrimination process. We observed the gradual increase

in activation around AU's during emotion expressions. The triplet network additional branch encourages the network to identify the extent of the object as compared to the Siamese network case which encourages it to identify less generalization of the transition. The triplet network shows less loss during training and smoother intensity estimation compared to Siamese network. Experiment details are mentioned in next section.

We utilized class activation to visualize the gradual change of activation around AU's. Lets say we are visualizing expression E intensity, activations visualizing of convolution units shows significantly higher values around AU's that is related to that emotion as shown in figure 4.3. Class activation map is designed to visualize one class at a time. Considering our problem isn't a classification problem, this requires altering the class activation to work for our ranking model. In our work, we consider the single output of a Siamese branch as a single class. This holds the concept of activation mapping and results in similar output of the standard method. Activations with low values (> 0.015) are ignored during the visualizing as suggested on [129]. When either networks encounters an apex, it shows the highest activations around regions that represent that emotion. For instance, highest activation is shown at the lips when estimating smile intensity while it looks both at eyebrows and mouth when classifying anger. This can be observed in figure 4.3 where an emotional transition to anger is taking place until apex is reached (last item in the sequence). The results match the findings of [147] who used same method to visualize class activation for emotion classification in pictures.

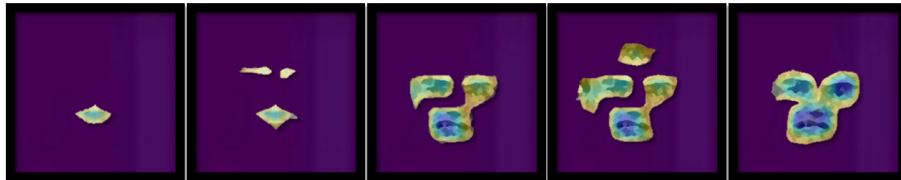


Fig. 4.3 This figure shows the class activation map for anger emotional transition. The incremental change in activation value and size is an indicates apex is occurring.

We also visualized feature maps of representations extracted from the connection between the second max pooling and the fully connected layer. Convolutional networks see inputs through windows with different sizes and move those windows through the image resulting in features as described by [145]. Feature maps capture those features and help understanding how the network is building it's knowledge. Figure 4.4 shows the emotional transition from onset to surprise apex to offset. Brighter colors illustrate regions with high activation and therefore higher importance in the decision making. In every emotional transition, we visualize unit outputs with the highest activations. Such units have dominant effect in

decision making by supplying those high activations to the loss function. Visualizations and analysis are discussed in the experiments section.

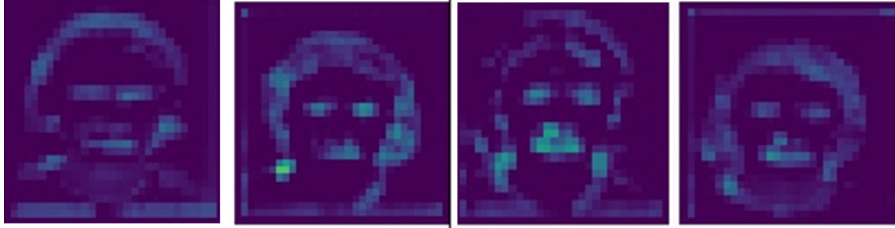


Fig. 4.4 This figure shows feature maps extracted from connection between second max pooling and the fully connected layer for surprise emotional transition.

4.4 Experiments

In our experiment, we train and evaluate Siamese and triplet networks for ranking using CK, MUG and MMI datasets. We evaluate their abilities to estimate expression intensities. Beside loss as an important measure of training efficiency, we analyze feature localization. Every emotional transition requires extracting a set of related features to get its intensity estimated. We show Siamese and triplet networks are efficient for such task. In the Siamese network, two branches observe the gradual change of expression in a sequence of images. In the triplet network, three branches are used in which anchor, negative and positive images are given to network. Anchor is the base image; the image is called positive if it's one step forward from the anchor. It's called negative if it's one step before the anchor.

4.4.1 Dataset

For three datasets: CK, MUG and MMI, each sequence begins with some neutral expression and proceeds to a peak expression. Subjects expressed the emotion that was to be expressed while seated three meters from the camera. The Extended Cohn-Kande Dataset includes 592 colored and gray-scaled sequences from 123 posers. The extended Cohn-Kande dataset comes with different angles describing 7 emotional states. The considered emotions are happiness, surprise, neutral, fear, sadness, disgust and anger. We specifically analyzed the improvement in intensity estimation and localization for both Siamese and triplet network. We generated pairs and triplets as mentioned previously in the data setup section. The number of generated triplets for CK dataset is 17343, 10405 items were used for training while 6938 items were used for testing. For Siamese network, number of generated pairs is 19503

pairs, 11700 for training and 7803 for testing. A sample of the unpaired data is shown in figure 4.5.



Fig. 4.5 This Figure shows the different emotions given by Cohen Kanade extended dataset. The seven emotional states are shown in random order. The states are happiness, surprise, fear, sadness, disgust, anger and neutral.

The MUG dataset consists of image sequences of 86 subjects performing facial expressions. The number of the available sequences counts up to 1462 with each sequence containing 50 to 160 images. subjects are adult men and women. Men are with or without beards. There are no occlusions except for a few hair falling on the face. The considered emotions are happiness, surprise, fear, sadness, disgust and anger. In a similar experiment to CK dataset, we analyzed the improvement in intensity estimation and localization for both Siamese and triplet network. we generated 19600 triplets, 11760 items were used for training while 7840 items were used for testing. For Siamese network, number of generated pairs is 20100 pairs, 12060 for training and 8040 for testing. A sample of the unpaired data is shown in figure 4.6.

The MMI dataset consists of 2900 videos of 75 subjects. It is annotated for the presence of AUs in videos indicating for each frame whether an AU is in either the neutral, onset, apex or offset phase. The emotions in the videos are happiness, surprise, neutral, fear, sadness, disgust and anger. We estimated the intensity and localization for both Siamese and triplet network for this dataset. We generated pairs and triplets as mentioned previously in the data setup section. The number of generated triplets for MMI dataset is 27634, 16580 items were used for training while 11054 items were used for testing. For Siamese network, number of generated pairs is 29409 pairs, 17645 for training and 11764 for testing. Figure 4.7 shows a sample.



Fig. 4.6 This Figure shows the different emotions given by MUG dataset. The 6 emotional states are shown in random order. The states are happiness, surprise, fear, sadness, disgust, anger.

For the Micro-expressions experiment we used CASME, CASME II and CAS(ME)² datasets. CASME has 195 micro-expressions recorded under 60 fps. Its samples were coded so that the beginning and the ending of the Micro-expression is tagged. CASME II dataset has 247 micro-expressions with AUs labeled. Baseline frames are usually neutral and are kept before and after each micro-expression. The recordings were taken under 200 fps. CAS(ME)² dataset contains macro-expressions and micro-expressions samples that were collected from the same participants in the same experimental conditions. We generated pairs and triplets as mentioned previously in the data setup section. Sample of unpaired data is shown in 4.8.

4.4.2 Evaluation measures

The first and last frame of each Cohen-Kanade datasets sequence are onset and apex frame respectively. This defines the minimum and maximum relative intensity of each sequence. While MUG and MMI datasets sequences start from onset then apex then close to onset again. The distance between onset and apex varies from person to another so we can compare intensity values across different subjects. To evaluate the performance of intensity estimation,



Fig. 4.7 This Figure shows the different emotions given by MMI dataset. Variety of emotions is shown in random order. The states are happiness, surprise, fear, sadness, disgust, anger.

we use the artificial approach given in [170]. The intensity within the pair y_k varies between $I_l = 0$ and $I_h = 1$ and is given using it's relative distance to the corresponding apex frame p of the sequence as shown in (3):

$$y_k = \frac{k-1}{p-1}(I_h - I_l)\delta_{k < p} + \frac{m-k}{m-p}(I_h - I_l)\delta_{k \geq p} \quad (4.3)$$

where δ is the indicator notation and $k= 1, \dots, m$ and m is the length of the sequence. Therefore y_k is an intensity curve that is found to change linearly between onset and apex frames or from onset and apex frames. We calculate y_k per frame in sequence and refer to it as our ground truth. Sequences with higher frame rate result in longer ground truth when compared to sequences of the same duration. The length of every sequence matches the length of the corresponding ground truth of that sequence. Lets take the sequence in figure 4.9 as an example. For all sequences, $I_l = 0$ and $I_h = 1$. In this case, the length of the



Fig. 4.8 This Figure shows the different samples given by CASME, CASME II and CAS(ME)² datasets. The frames that define the beginning, peak and ending of the Micro-expression is annotated.

sequence is $m = 8$. The apex frame in this case is manually annotated to be $p = 4$. Therefore for all values of y_k where $k < p$, the first term of equation 3 is used such as $y_2 = \frac{1}{3}$. On the other hand, if $k \geq p$ then y_k equals to the second term of equation 3 such as $y_7 = \frac{1}{4}$.

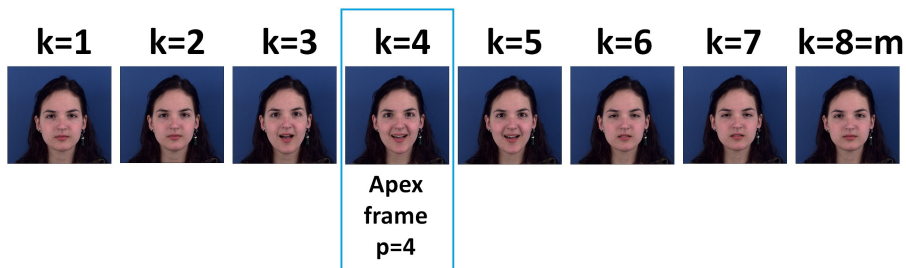


Fig. 4.9 Figure shows an example of how the ground truth of an emotional transition is calculated. This frame sequence is simplified and has 8 items. y_k is calculated per each item k .

We used Pearson correlation coefficient (PCC), intra-class correlation (ICC) and mean absolute error (MAE) to evaluate the estimated intensity and its relation to the ground truth. PCC measures how well the prediction can capture the trend of intensity change. ICC measures the consistency within each intensity level. By definition, PCC and ICC range between 0 and 1 with larger being better. MAE measures the deviation between prediction and actual value. It results in a positive value with smaller being better. For comparison, we use OSVR-L1, OSVR-L2 mentioned in [170], Rankboost mentioned in [166] and OR mentioned by [155]. OSVR-L1 and OSVR-L2 are trained given relative intensity of key frames.

Rankboost and OR are trained using ordinal information only. For evaluation, PCC, ICC and MAE are used for all the methods. The results of two networks are shown in next sections.

4.4.3 Experiment 1- CK Dataset:

Siamese Network

Our network receives images with 224×224 pixels which defines the input size of the network. We evaluate the network loss and accuracy compared to the calculated ground truth beside the localization ability. Location of the feature and the high activations around important regions are visualized. Intensity estimation is evaluated through charts considering two levels of engagement per emotion (higher and lower). The training took 100 Epochs with a mini-batch size of 20. The loss function for both networks training is the exponential loss and loss chart is shown in figure 4.10.

In our Siamese network experiment on CK dataset, we used 11700 pairs for training and 7803 pairs for testing. We used SGD optimizer with learning rate 0.001 without weight decays. Estimation results shown in table 1. We refer to our Siamese network as SIE. The work based on extracting ordinal information done by [170] is referenced as OSVR-L1 and OSVR-L2. The work that uses large margin rank boundaries for ordinal regression is referenced as OR. Rankboost is referenced by it's name. Those methods are considered the state of the art in the field of emotion intensity.

Triplet Network

With a similar structure to Siamese network, in our triplet network experiment on CK dataset we used 10405 triplets for training and 6938 triplets for testing. We used SGD optimizer with learning rate 0.001 without weight decays. We refer to our triplet model as TRIE. The estimation results are shown in last row of table 1.

Table 1 The results of our experiments on CK dataset for Siamese (SIE) and triplet (TRIE) networks beside RankBoost by [166] work denoted by it's name and [155] work denoted as OR and work of [155] denoted as OSVR-L1 and OSVR-L2. All methods are tested using same dataset and evaluated via PCC, ICC and MAE against the calculated ground truth.

Method	PCC	ICC	MAE
Rankboost	0.4115	0.3812	1.0670
OR	0.4741	0.4531	2.0856
OSVR-L1	0.4825	0.4011	2.6384
OSVR-L2	0.5411	0.4001	1.1179
SIE	0.650	0.576	0.229
TRIE	0.86	0.84	0.119

4.4.4 Experiment 2- MUG Dataset:

Siamese Network

Similar to our previous experiments, our input images dimensions are 224×224 pixels. We consider the network loss and accuracy compared to the ground truth beside the localization ability for evaluation. The training loss and intensity estimation are evaluated. The training took 100 Epochs with a mini-batch size of 20. The loss function for both networks training is the exponential loss and is the loss charts are shown in figure 4.11.

In our Siamese network experiment on MUG dataset, we used 12060 pairs for training and 8040. We used SGD optimizer with learning rate 0.001 without weight decays. Estimation results shown in table 2. We use the same references for our work and the state of the art methods as used in the previous experiments.

Triplet Network

With a similar structure to Siamese network, in our triplet network experiment on MUG dataset we used 11760 pairs for training and 7840 pairs for testing. We used SGD optimizer with learning rate 0.001 without weight decays. Estimation results shown in last row of table 2.

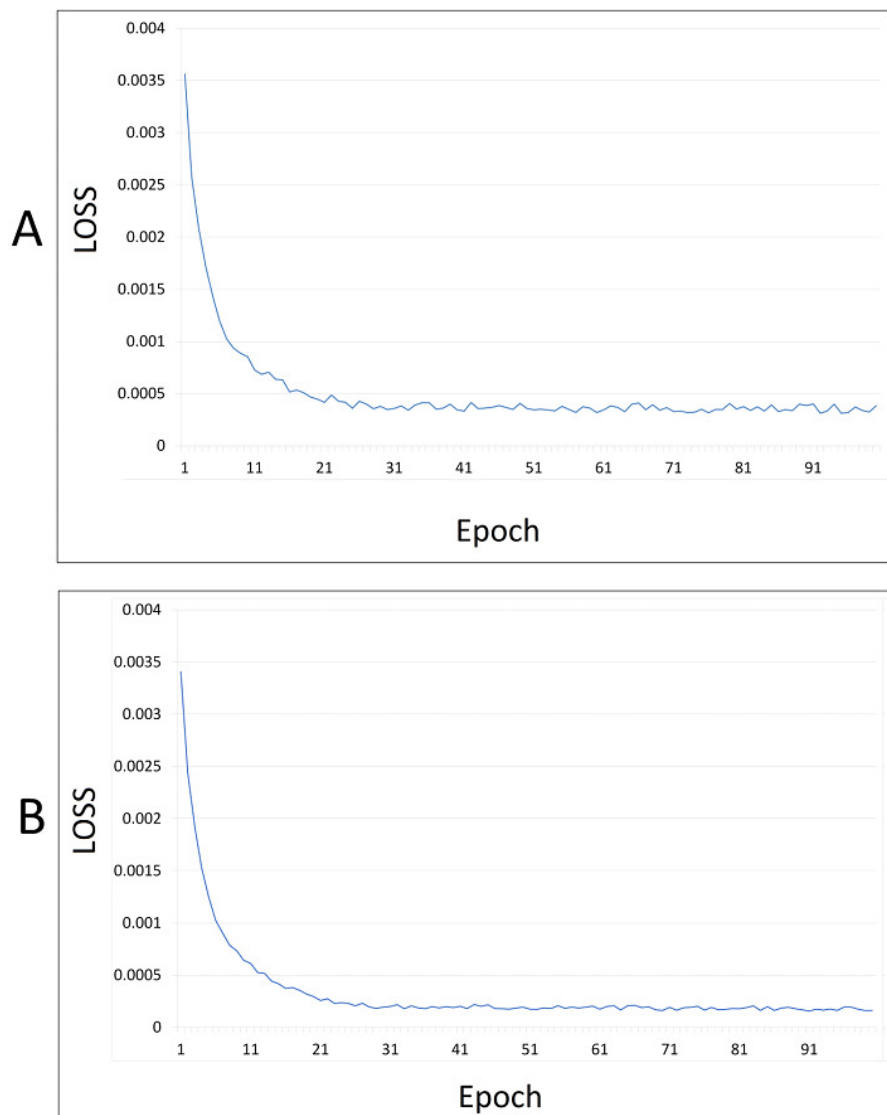


Fig. 4.10 This Figure shows the Loss during the training of Siamese (top) and triplet networks (bottom) for CK dataset. The triplet network overcomes the Siamese network by faster convergence and lower loss.

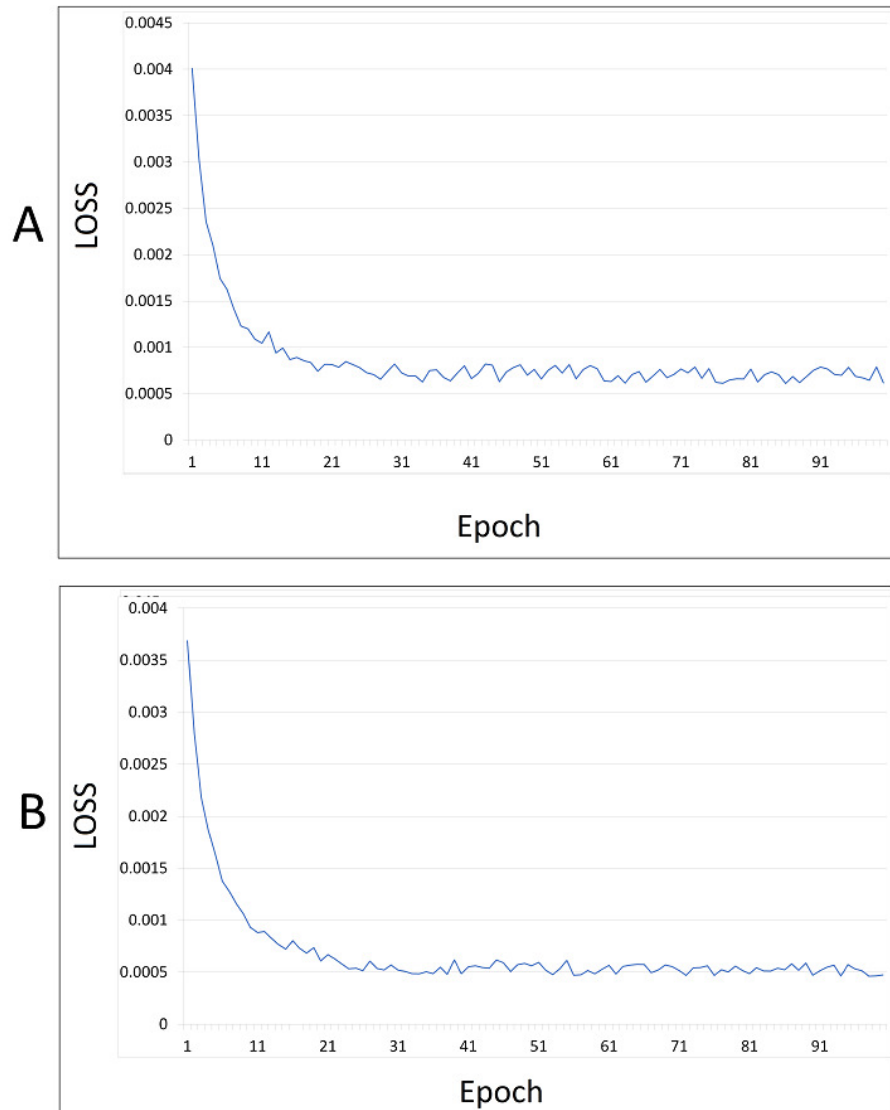


Fig. 4.11 This Figure shows the Loss during the training of Siamese (top) and triplet networks (bottom) for MUG dataset. The triplet network overcomes the Siamese network by faster convergence and lower loss.

Table 2 The results of our experiments on MUG dataset for Siamese (SIE) and triplet (TRIE) networks are shown against the state of the art methods. All methods are tested using same dataset and evaluated via PCC, ICC and MAE against the calculated ground truth.

Method	PCC	ICC	MAE
Rankboost	0.3912	0.3424	1.1975
OR	0.4122	0.3897	2.6012
OSVR-L1	0.4511	0.3496	2.7953
OSVR-L2	0.5111	0.3723	1.3948
SIE	0.5910	0.5296	0.549
TRIE	0.8129	0.7912	0.329

4.4.5 Experiment 3- MMI Dataset:

Siamese Network

Following the same setup of previous experiments, our input images dimensions are 224×224 pixels. We consider the network loss and accuracy compared to the ground truth beside the localization ability for evaluation. We evaluated training loss and intensity estimation. The training took 100 Epochs with a mini-batch size of 20. Exponential loss is used for networks training and loss charts are shown in figure 4.12.

In our Siamese network experiment on MMI dataset, we used 16580 pairs for training and 11054 for testing. We used SGD optimizer with learning rate 0.001 without weight decays. Estimation results shown in table 3. We use the same references for our work and the state of the art methods as used in the previous experiments.

Triplet Network

With a similar structure to Siamese network, in our triplet network experiment on MMI dataset we used 17645 pairs for training and 11764 pairs for testing. We used SGD optimizer with learning rate 0.001 without weight decays. Estimation results shown in last row of table 3.

4.4.6 Experiment 4- Micro-Expressions:

In this experiment we evaluate the Siamese and triplet network abilities to detect Micro-Expressions in CASME, CASME II and CAS(ME)² datasets. Following the same setup of previous experiments, our input images dimensions are 224×224 pixels. We consider the network localization ability for evaluation while our models are designed for classification. Both datasets have annotated the index of frames in which Micro-Expressions appear, peak

Table 3 The results of our experiments on MMI dataset for Siamese (SIE) and triplet (TRIE) networks are shown against the state of the art methods. All methods are tested using same dataset and evaluated via PCC, ICC and MAE against the calculated ground truth.

Method	PCC	ICC	MAE
Rankboost	0.4008	0.3501	1.2199
OR	0.4296	0.3971	2.7902
OSVR-L1	0.4596	0.3596	2.8314
OSVR-L2	0.5209	0.3914	1.4216
SIE	0.5973	0.5596	0.699
TRIE	0.7996	0.8011	0.399

up and disappear. We use this information to evaluate detection of Micro-Expressions by Siamese and triplet models.

Siamese Network

In our Siamese network experiment on CASME, CASME II and CAS(ME)² datasets, we used 4515 pairs for training and 950 for testing. We used SGD optimizer with learning rate 0.001 without weight decays. Given that the annotated frames are the ground truth, we evaluate the presence and peaking up of Micro-Expressions during the annotated period as shown in figure 4.13.

Triplet Network

In our triplet network experiment on CASME, CASME II and CAS(ME)² datasets we used 5519 pairs for training and 1109 for testing with similar setup to the Siamese experiment. It's difficult to give quantitative amount for our evaluation measure. Therefore findings shown in figures 4.13 and 4.14 are discussed in next section.

4.5 Discussion

In this section we discuss the results of Siamese and triplet networks on CK, MUG and MMI datasets for intensity estimation and CASME, CASME II and CAS(ME)² datasets for Micro-Expressions detection.

In figure 4.15, we visualized the Siamese network class activation on CK dataset. The left most sequence for instance, is evaluating surprise. The network activation maps are built up gradually (bottom photo sequence) to show eyebrows being raised, eyes are widened and lower lip is moved downward during the transition. This matches the definition of surprise

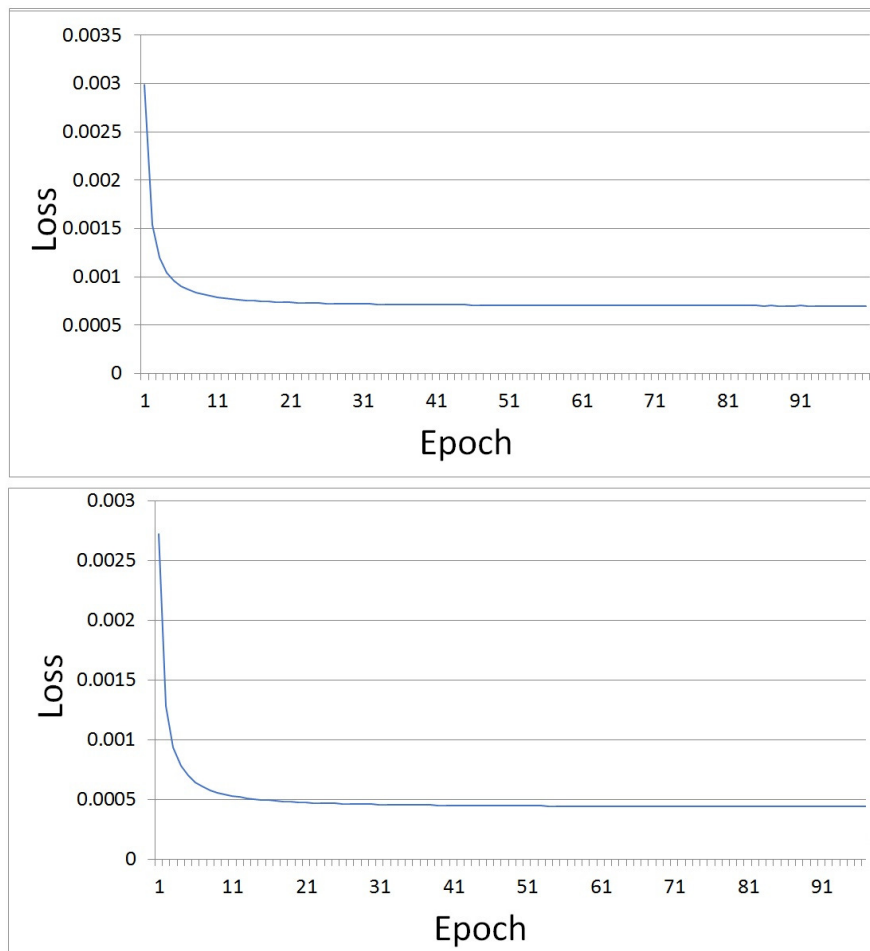


Fig. 4.12 This Figure shows the Loss during the training of Siamese (top) and triplet networks (bottom) for MMI dataset. The triplet network overcomes the Siamese network by faster convergence and lower loss.

given by FACS [151]. Second sequence from left is showing similar gradual build up but with higher intensity. This explains why the activations are more condensed and the intensity chart is showing high value. The third and fourth sequences are showing 2 cases of intensity estimation for sadness generally described by dropping upper eyelids and slight pulling down the lips.

We also visualized the class activation of the triplet network on CK dataset. Figure 4.16 shows how the internal representation of features improves by using triplet network. The left most sequence of figure 4.16 for instance, is evaluating happiness with wrinkling crows feet, pushing up cheeks and muscles that orbits the eye move during the transition. Class activation maps gradually highlight those action units during the transition. The second sequence in figure 4.16 is showing similar gradual build up but with slightly higher happiness

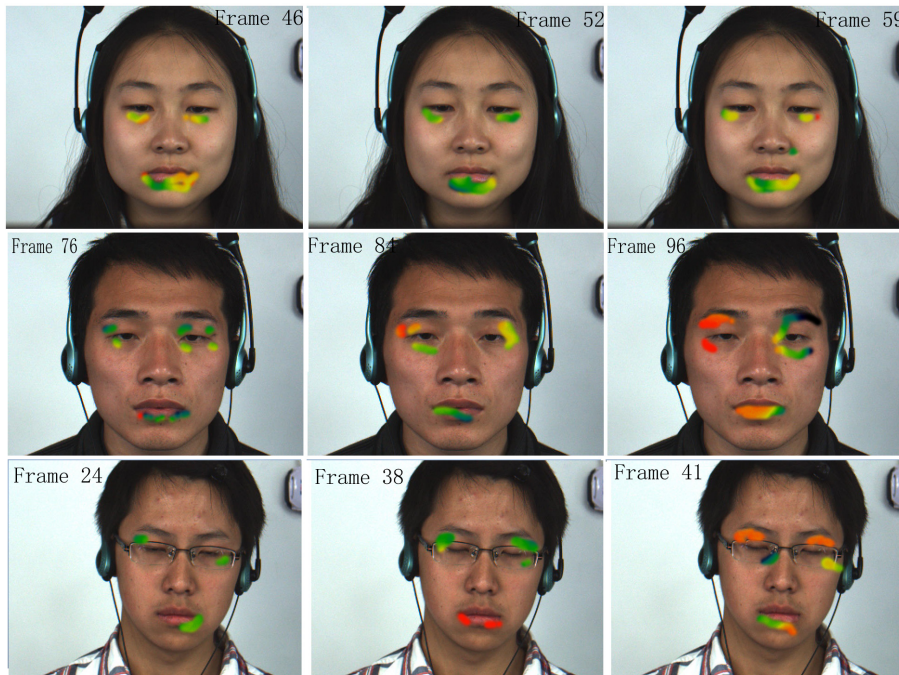


Fig. 4.13 This figure shows the localization of class activation for Micro-Expressions occurrences in CASME, CASME II and CAS(ME)² datasets using Siamese network. Emotions in this figure (from top) are happiness, fear and sadness. Due to the length of the sequences, we only included ordered pictures.

engagement reflected by the class activation map and the intensity chart. The third and fourth sequences of figure 4.16 are showing 2 cases of intensity estimation for disgust generally described through activation maps by wrinkling nose and raising the upper lip. All those movements match the definition of surprise given by FACS ([151]). It is observed that the triplet network is showing smoother intensity charts. The additional branch of triplet network helps collecting information about the negative component of the triplet. This ensures the network will focus on the details correctly describing the emotional transition.

Both models show proper localization for AU's that reflect certain expression. The localization abilities of triplet network significantly become restricted towards more critical regions. This is observed via feature map in figure 4.17. Considering surprise for instance in 4.17-A, the network showed blur surprise facial components during the transition to give its prediction. While 4.17-B shows higher activations around mouth, nose and eyes when using triplet network.

In figure 4.18, we visualized the class activation of the Siamese network for MUG dataset. The left most sequence of figure 4.18 for instance, is evaluating surprise showing AU's sequential change similar to the CK dataset. The second sequence is showing similar gradual

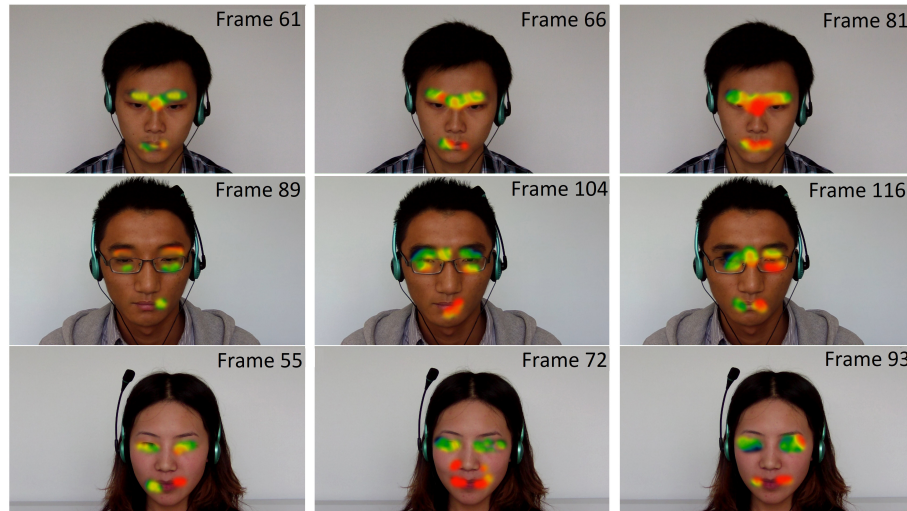


Fig. 4.14 This figure shows the localization of class activation for Micro-Expressions occurrences in CASME, CASME II and CAS(ME)² datasets using triplet network. Emotions (from top) are disgust, fear and happiness.

change but with higher intensity. Those higher activations in the second sequence are key to describe higher intensity than the first sequence. The third and fourth sequences are showing 2 different intensity estimations for fear. Class activation maps are showing eyebrows being pulled together, raising upper eyelids, tensing lower eyelids and stretching lips slightly horizontally. Emotional transition in MUG dataset goes from onset to Apex to onset again. The intensity estimation starts from a low value and gradually increasing until apex is reached and gradually degrading until reaching the onset again. Not all transitions ending onset is similar to the first onset. The reason can be clearly observed through transitions like third sequence in figure 4.18 in which the last onset still contains details of the apex (eyebrows being pulled together). The results are reflected on intensity charts.

We visualized the class activation of the triplet network on MUG dataset. Figure 4.19 shows how the internal representation of features improves by using triplet network. The left most sequence in figure 4.19 for instance, is evaluating sadness generally described by dropping upper eyelids and slight pulling down the lips during the transition. Class activation maps are shown gradually precisely around those regions. The second sequence in figure 4.19 is showing similar gradual change but with slightly higher sadness engagement reflected by the class activation map and the intensity chart. The third and fourth sequences are showing 2 cases of intensity estimation for anger. Activation maps appear around eyes showing glare, lips showing narrowing and eyebrows region appearing down and together. All those movements match the definition of surprise given by FACS [151]. The triplet intensity chart is smoother than Siamese intensity chart and has less spikes.

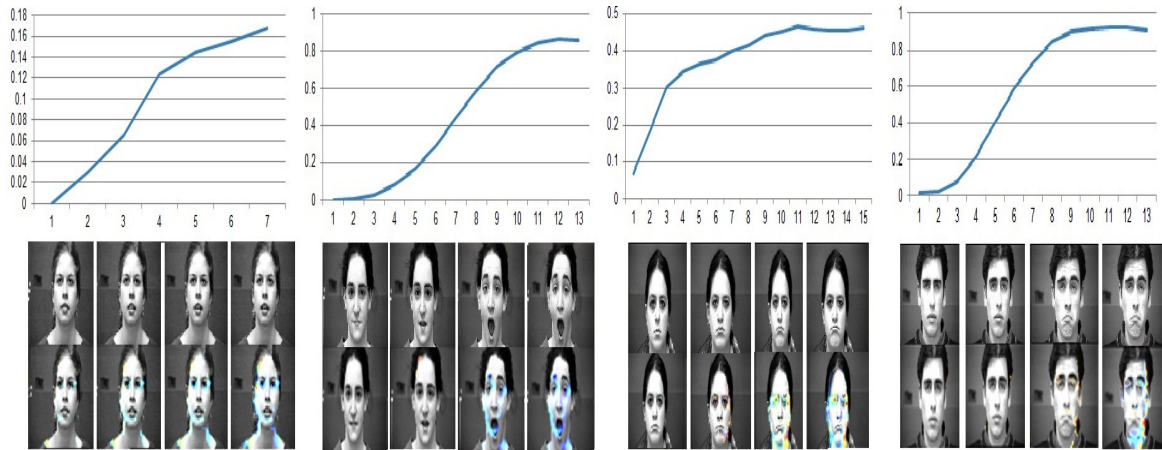


Fig. 4.15 This Figure shows the intensity estimation using Siamese network for CK dataset. low and High engagements of surprise and sadness is shown from left to right.

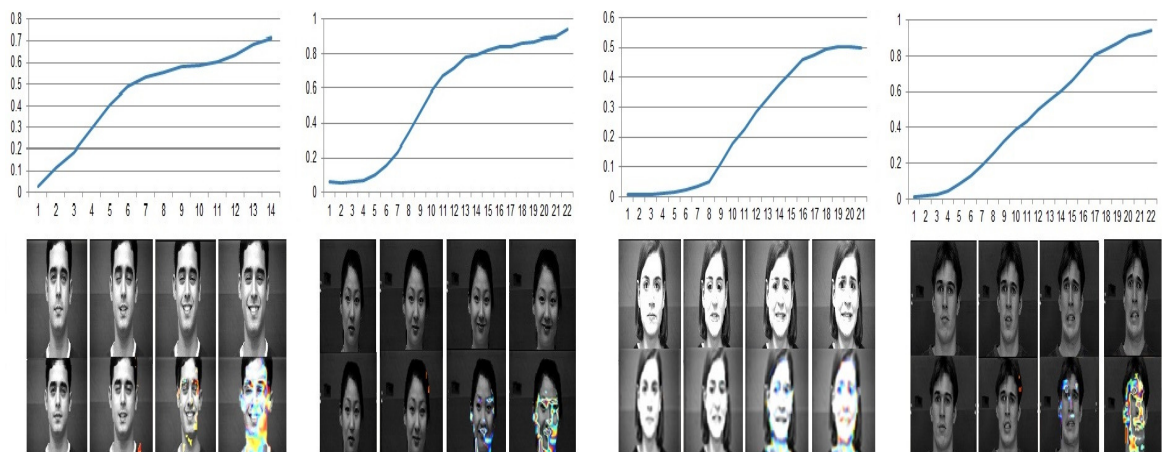


Fig. 4.16 This Figure shows the intensity estimation using Triplet network for CK dataset. low and High engagements of happiness and disgust is shown from left to right.



Fig. 4.17 This Figure shows the evolution of feature map during the surprise emotion transition. Saliency is key when determining the efficiency of a feature map representation over the other. The feature maps of Siamese network is shown in (A) while (B) shows the triplet feature maps. Clearer features are extracted through triplet networks when compared to Siamese.

In figure 4.20, we visualized the class activation of the Siamese network for MMI dataset. The left most sequence of figure 4.20 for instance, is evaluating happiness with wrinkling crows feet, pushing up cheeks and muscles that orbits the eye move during the transition similar to the CK dataset. The second sequence is showing similar gradual build up but with slightly higher happiness engagement reflected by the class activation map and the intensity chart. Those higher activations in the second sequence are key to describe higher intensity than the first sequence. The third and fourth sequences are showing 2 different intensity estimations for surprise. Class activation maps are showing eyebrows being raised, eyes are widened and lower lip is moved downward during the transition.

We visualized the class activation of the triplet network on MMI dataset. Figure 4.21 shows how the internal representation of features improves by using triplet network. The left most sequence in figure 4.21 for instance, is evaluating disgust generally described by wrinkling nose and raising the upper lip. Class activation maps are shown gradually precisely around those regions. The second sequence in figure 4.21 is showing similar gradual change but with slightly higher disgust engagement reflected by the class activation map and the intensity chart. The third and fourth sequences are showing 2 cases of intensity estimation for fear. Activation maps appear showing eyebrows being raised and pulled together, upper eyelids being raised, and lower eyelids being tensed. All those movements match the definition of fear given by FACS [151]. The triplet intensity chart is smoother than Siamese intensity chart and has less spikes. Emotional transition in MMI dataset goes from onset to Apex to onset again. The intensity estimation starts from a low value and gradually

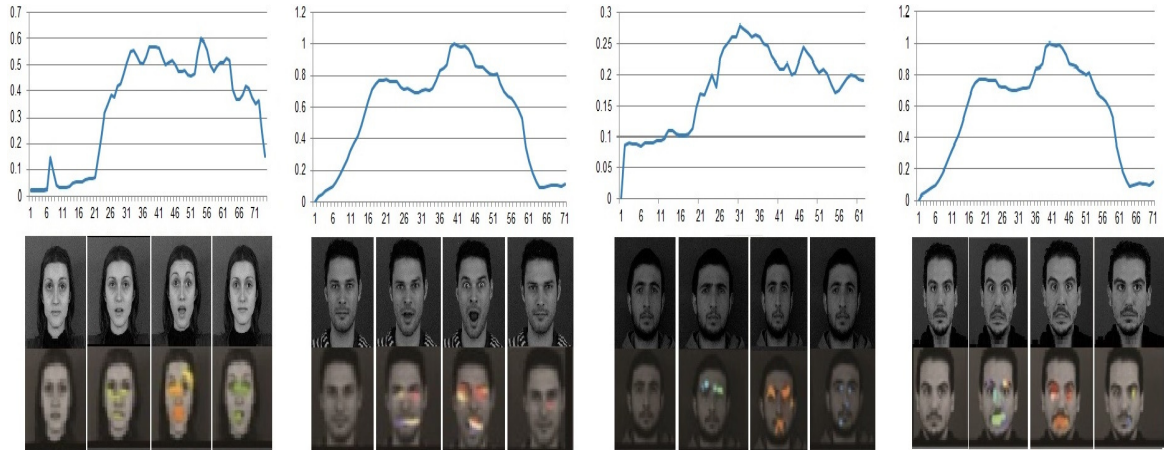


Fig. 4.18 This Figure shows the intensity estimation using Siamese network for MUG dataset. Each sequence shows the intensity change within the transition. The sequence on top contains the transition while the ones below is the masked corresponding activation. For demonstration, we only included ordered pictures with highest representation.

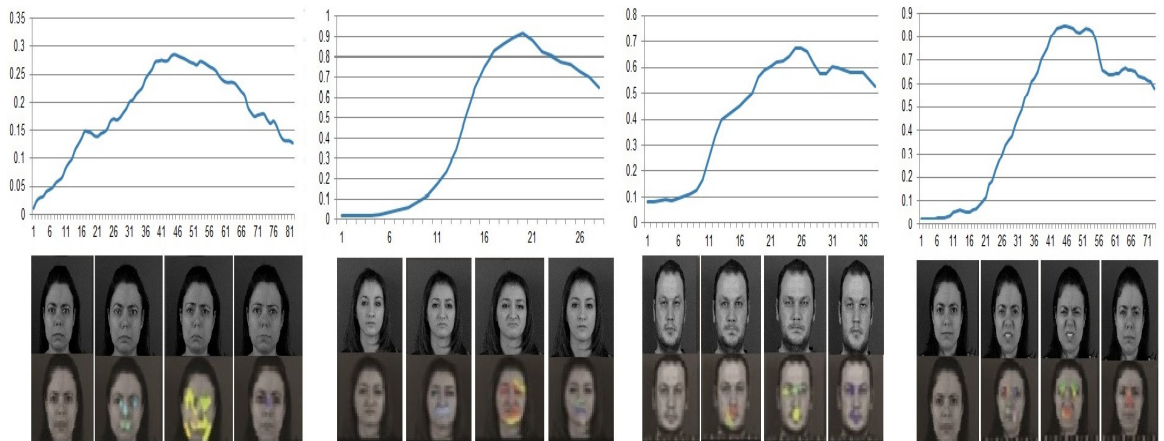


Fig. 4.19 This Figure shows the intensity estimation using Triplet network for MUG dataset. Triplet network intensity estimation has less spikes when compared to Siamese chart.

increasing until apex is reached and gradually degrading until reaching the onset again. Not all transitions ending onset is similar to the first onset. The reason can be clearly observed through transitions like third sequence in figure 4.21 in which the last onset still contains details of the apex (eyebrows being pulled together). The results are reflected on intensity charts.

Figures 4.13 and 4.14 show a mixture of Micro-Expressions that appear beside more lasting expressions. When Siamese and triplet models are trained to minimize the loss they gain ability to find salience changes between inputs. In figure 4.13, the network is able to find Micro-Expressions of happiness (top row) by detecting the raising cheeks and pulling lip corner during the annotated time window. Our model also detected fear in the second sequence in figure 4.13 by highlighting the movement of raising upper eyelids and tensing lower eyelids and slightly stretching lips. The Siamese network also detected sadness in the last sequence in figure 4.13 by highlighting the slight movement of upper eyelids and pulling down the lips during the transition. The results match the findings of [163] and [164] in terms of the location and the frames that Micro-Expressions occurred at.

In figure 4.14, the network is able to find Micro-Expressions of disgust (top row) by the activation increase around two regions due to wrinkling nose and raising the upper lip. Our model also detected fear in the second sequence in figure 4.14 through highlighting the movement at upper eyelids and tensing lower eyelids and slightly stretching lips. The triplet network also detected happiness in the last sequence in figure 4.14 by highlighting the raising of cheeks and pulling lip corner during the annotated time window. Those results match the annotations mentioned in both [163] and [164].

Activation location reflects the regions that the models use to conduct their task and the value of the activation reflects how important is this region for the decision making. It is observed from figures 4.13 and 4.14 that the triplet network collected more information than the Siamese network. This is predictable due to the extra branch that the triplet network is equipped with which gives longer temporal window. Siamese and triplet models didn't only detect Micro-Expressions but also adjacent regions that play important role in estimating emotion intensity.

4.6 Conclusion

[ht] In this chapter, we investigated the abilities of Siamese and triplet networks to detect Micro-Expressions and estimate emotional intensity in temporal domain by creating data pairs and triplets from sequenced emotional transition. We proposed detailed explanation through class activation maps representation on how our models are able to estimate intensity

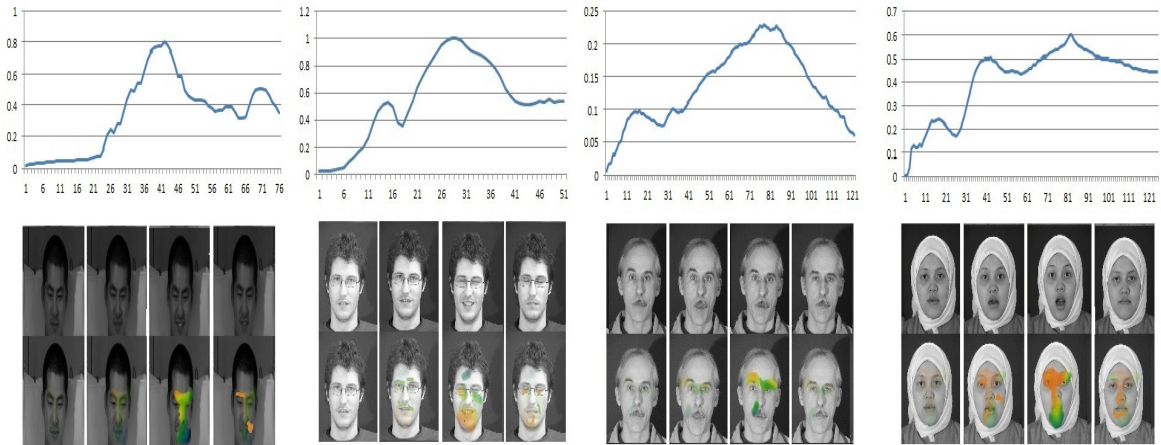


Fig. 4.20 This Figure shows the intensity estimation using Triplet network for MMI dataset. Each chart shows the intensity change within the sequence. The first row shows the natural evolution of an emotion. The second row shows the class activation mapping and the evolution of the emotion in the temporal domain.

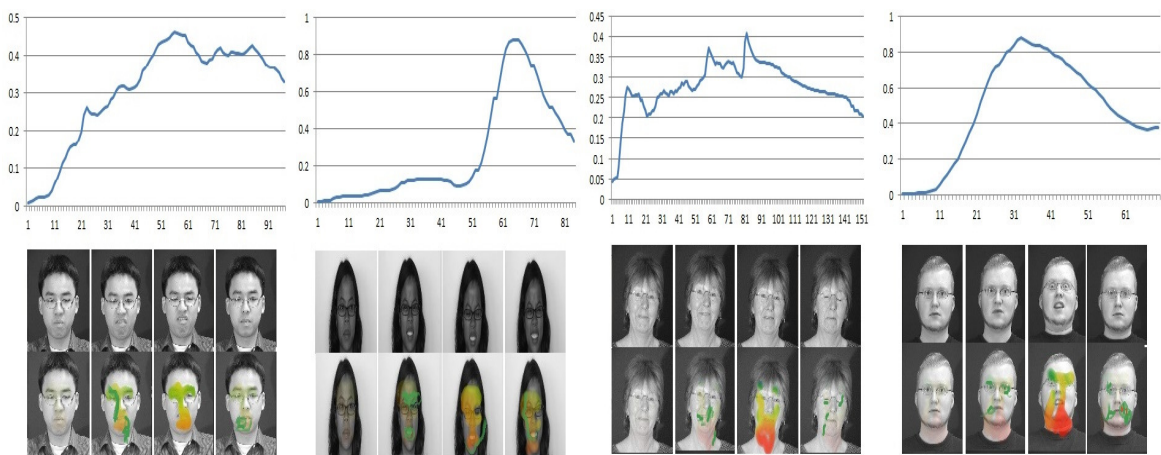


Fig. 4.21 This Figure shows the intensity estimation using Triplet network for MMI dataset. Triplet network intensity estimation has less spikes when compared to Siamese chart.

change by gradually highlighting action units through an emotional transition. Trained models have learned to perform object localization and highlighting the discriminative AU's without using any bounding box annotations. The proposed models were evaluated against multiple datasets and steadily exceeded in accuracy of emotional intensity estimation and also succeeded in detecting Micro-Expressions beside other longer lasting expressions. Siamese and Triplet networks have significant ability to sense an emotional transition at its very early stage. Many applications benefit from it specially in medication offering more comforting treatment or in education helping reticent students overcoming stressful lessons.

Next chapter discusses another aspect of emotion understanding which is the eye movement. the problem of gaze-tracking and blink-detection under illumination variation and unrestricted subject movement is addressed. A method of gaze direction tracking in real time video stream is proposed.

Chapter 5

Dynamic eye tracking through Gabor filters

Gaze tracking system tracks the gaze movement and detects concentration point. It offers a rich source of information for applications that utilize gaze tracking data. Those applications serve for medical diagnoses and human-computer interface. Gaze tracking is a very important media to deliver commands to machines through the eyes, especially in wearable computers.

Historically and according to surveys, gaze tracking is conducted through numerous approaches such as the electro-oculography approach and the video-oculography approach [99]. The electro-oculography approach requires attachments or plantation of devices into human eyes. This limits the long term usability and causes a disturbance when it comes to everyday activities. The video-oculography approach utilizes image processing techniques to track gaze without physical attachment with the user. This property allows this approach to be widely adopted in many applications and researches. Many researchers enhance the input media directly or indirectly through software and hardware solutions. The Hardware solutions such as exposing the subject to IR light result in efficient gaze direction tracking but have many reported health effects especially when encountered for a long time [100, 101]. This limits the current tracker's integrations in many of everyday applications [99]. Software solutions achieve a significant breakthrough in gaze direction tracking. On the other hand, those solutions tend to be sensitive to lighting change and subject movement [100, 101].

In this chapter, we propose a technique that overcomes these limitations in tracking gaze direction. Our technique preserves tracking in high accuracy of unsteady subjects under lighting change without using additional hardware. We show our model in Figure 5.1. Initially, the region of interest is constructed to minimize processing window, eliminate most of the background noise and offer unrestricted head movement. The region of interest is established by detecting the face of the subject and generating a template for the eyes region to be traced and processed during the tracking time. Gabor filter is applied for the region of interest afterwards. Because of the nature of the Gabor function, most of the lighting

variations are eliminated. This offers illumination change resistance while tracking the gaze. We track the gaze direction after the Gabor filter has highlighted the gaze part of the eye. Considering that most pupil center estimation algorithm (PCEA) complexity equals to or exceeds $O((m \times n)^2)$ [102, 103] where m and n are the dimensions of the processing area, shrinking the processing area boosts our technique efficiency greatly.

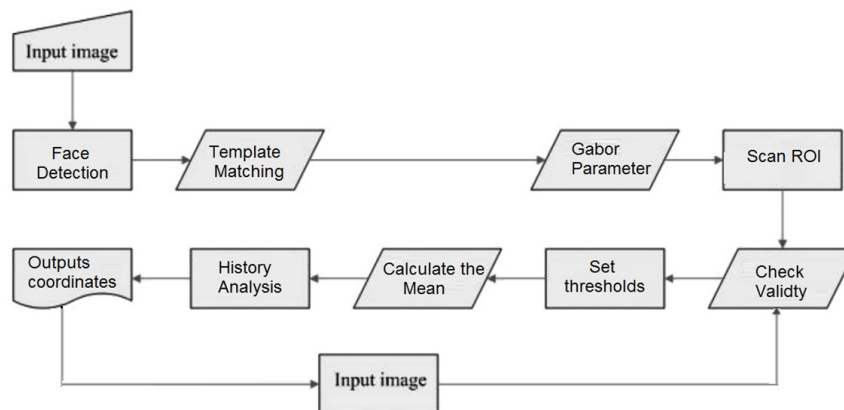


Fig. 5.1 Modular description for the proposed technique shows the stages of frames processing.

The Gabor Filter and its function's second order matrix are used to identify edges efficiently. Inequality of the matrix Eigenvalues with local intensity-based threshold determines the points where illumination changes. The threshold is set automatically aiming towards making gaze edge visible within the region of interest. The group of points in which the intensity is changing forms the gaze edge. Validity checks for the detected edges are performed to ensure threshold is not producing unwanted edges within the eye region.

Our technique offers reliability and accuracy that exceeds those of the state of the art methods. Our method also offers an ability to detect blinking which is considered a challenge in many recent gaze trackers [104]. The absence of one or both gazes is a sign of a blink. Yet shortly after its reappearing, our method continues gaze tracking.

The remaining parts of this chapter are organized as follows: In Section 2 some related work was highlighted. Our proposed techniques for gaze direction tracking and blink detection are presented in Section 3. In Section 4 comparisons with state of the art techniques are introduced. In Section 5, results for our work and the compared techniques are discussed. Finally, in section 6 we conclude our work.

5.1 Related Work

The video-oculography gaze trackers artificially alter the processing media to highlight minor yet useful details. Gaze centers can be tracked by three model types. The first one is based on geometrical features such as gaze shape[99]. This technique is useful for extraction via simple models such as Hough circularity model considering gaze circularity. Those models work in ideal cases in which the subject is frontally facing the camera. The second tracking model is based on optical features such as the corneal reflection due to infra-red light. It has been intensively used in gaze tracking due to its illumination variation immunity[99]. The third type is the appearance based approach. A template is used to trace certain object within the eye. It's scalable and able to endure rotation and occlusion of eyes [105, 99]. Two techniques following the appearance based approach allowing tracking under natural head movement were introduced by Zhu and Ji [106]. When the gaze is tracked, visual axis can be calculated from the optical axis by a simple calibration procedure. Generally speaking, systems that utilizes Image processing techniques tend to be more sensitive to lighting change within the processing media [112, 113].

Villanueva and Cabeza [107] proposed a geometric-model-based technique that utilizes one camera and multiple light-emitting diodes (LEDs) to track gaze. Introducing more devices adds complexity to any proposed methods. Li et al. developed a tracking system to detect the pupil features [102] that has been later applied into an eye-tracking system[111]. On the other hand, models that combine the facial pose and eye location lower the accuracy [108]. Hennessey et al. [109] and Countinho et al. [110] also confirmed this general phenomenon by their experiments respectively. Kumar et al. introduced a new algorithm for a novel head-mounted camera [114].

Inspired by the hybrid of dominant features and appearance based approaches and considering the limitations in hardware techniques we propose a gaze direction tracking model from a single camera. Our technique offers an accuracy that exceeds the state of the art techniques. This is proven through experiments in the section 4. In our approach, both Gabor filter and its function's second moment matrix are used to locate the gaze edges. The concentration points are then scanned and validated to ensure reliability. During the processing time the user can freely move and change body alignment while preserving high tracking accuracy. Moreover, we propose a solution for the eye blink issue in this chapter.

5.2 Proposed Methodology

Gazes are the circular boundaries of the eye ball that represent visual concentration point of the subject. It comes with distinguishable illumination compared to the rest of the eye. Based on it, relative positions of gaze illumination is extracted and analyzed. In order to extract the illuminating part, our gaze detector is developed to track the gaze illumination and its location relative to the face. Many edge detectors have been proposed to detect illumination or eye structure, of which Thomasi et. al.[115] and Harris Corner Detector [116] are the representative methods. However, the former one is limited to detect coroners with certain shapes and therefore limits subject movement freedom [117].

Due to those limitations, we introduce a different approach in our work based on Gabor Filter and its function's second moment matrix. We first construct the processing area that contains the eyes region. This step happens only once when the tracking starts for a new subject. The region of interest is constructed by firstly detecting the face of the subject by the Normalized Pixel Difference algorithm [118]. Later the eye region is cropped and a template is built and tracked by the cam-shift algorithm [119]. This region of interest contains rich and useful information for multi-resolution analysis of the signal behavior. The Gabor filter has the ability to extract multi-scales and orientations details from a signal. Aside from this useful property, the Gabor Transform to images offers immunity to minor lighting variations and movements [117].

A 2-D Gabor function is defined by two components. The first component is a 2-D Gaussian-shaped function, and the second part is a complex sinusoidal carrier.

$$G(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2} \left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right]\right) \exp(i2\pi f_0 x) \quad (5.1)$$

where σ_x and σ_y characterize the Gabor function's spatial extent and frequency bandwidth, respectively. f_0 is the filter's central frequency and i represents the imaginary notation. When the highest density of the local spectral energy is around certain frequency for a signal, the Gabor Transform become a special form of the Fourier transform. The Fourier transform of the 2-D Gabor function $G(x, y)$ is given by

$$\phi(u, v) = \exp\left(-\frac{1}{2} \left[\frac{(u - f_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2} \right]\right) \quad (5.2)$$

The variables u and v are the rectilinear coordinates in the transformed frequency domain. Given an image position at $I(x,y)$, the corresponding Gabor Transform is written as

$$T_{s,r}(x, y) = a^{-s} \sum_{\xi} \sum_{\eta} I(\xi, \eta) \phi(x' - \xi, y' - \eta) \quad (5.3)$$

where x' and y' are the rotated coordinates and are given as $x' = a^{-s}(x \cos \theta_r + y \sin \theta_r)$ and $y' = a^{-s}(-x \sin \theta_r + y \cos \theta_r)$ respectively. s and r are index parameters. a^{-s} is the scaling factor with s varies from zero until the total number of scales. θ_r refers to the anti-clock wise rotation with r varies from 0 to total number of orientations.

Keeping in mind that the region of interest is smaller than the image itself, lower processing time is needed to apply the Gabor filter to our sub-image [102, 103]. Applying Gabor filter to an image offers better performance for local feature extraction due to its multi-resolution and orientation properties [11, 13]. According to the idea from Harris detector, if a point belongs to a corner, the intensity around this point may change greatly [116]. The gaze itself has stronger intensity level compared to eye corners. Guided by this fact we create second moment matrix using the Gabor Transform function. Suppose this second moment matrix is denoted as

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad (5.4)$$

where

$$A = \sum_{x,y} \left(\frac{\partial T_{s,r}(x, y)}{\partial x} \right)^2, \quad (5.5)$$

$$B = C = \sum_{x,y} \left(\frac{\partial T_{s,r}(x, y)}{\partial x} \right) \left(\frac{\partial T_{s,r}(x, y)}{\partial y} \right), \quad (5.6)$$

$$D = \sum_{x,y} \left(\frac{\partial T_{s,r}(x, y)}{\partial y} \right)^2. \quad (5.7)$$

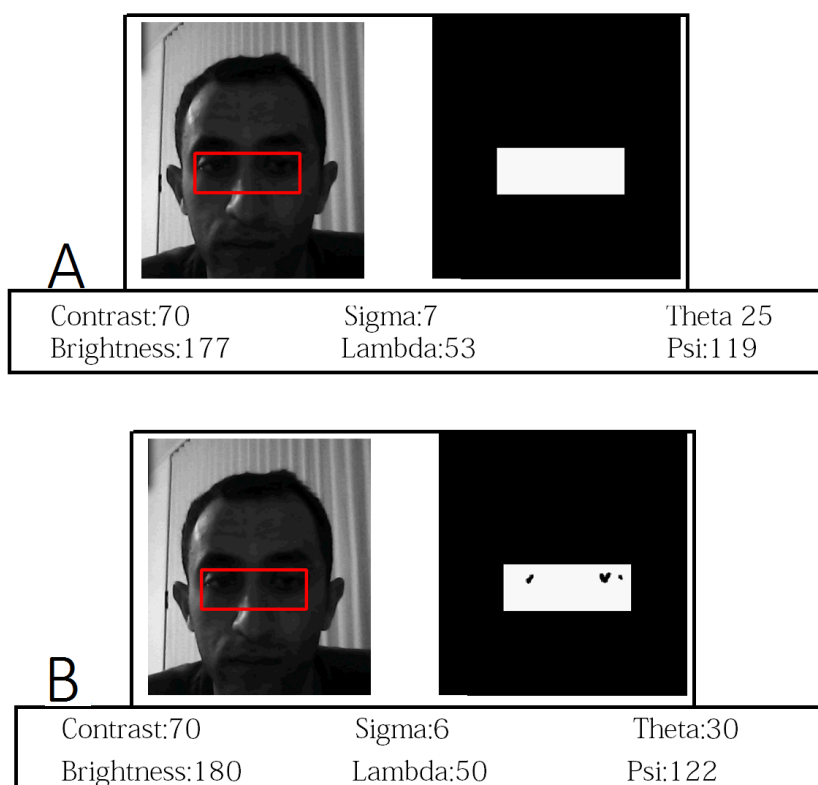
$\frac{\partial T_{s,r}(x, y)}{\partial x}$ and $\frac{\partial T_{s,r}(x, y)}{\partial y}$ are the first order partial derivatives of the Gabor Transform in the x and y direction respectively.

Matrix M defines the prevailing directions of the neighboring gradients and the amount to which these directions are coherent at certain point. We apply the gaze intensity determinate criterion of the higher transformed Gabor space. This criterion includes information about high intensity point and its neighborhood. The Eigen values of the second moment matrix

define the intensity. The Eigen values of M is given as

$$\lambda_{1,2} = \frac{(A+D) \pm \sqrt{(A-D)^2 + 4BC}}{2} \quad (5.8)$$

If the Eigenvalues are both larger than a given threshold, the intensity changes along any direction are large, which identifies the location of the gaze. If the threshold value is greater than one of the Eigenvalues and smaller than the other then the intensity changes significantly towards one direction at that point. This means current point refers to the outer edge of the eye. Finally, if both eigenvalues are below the threshold, no valuable information regarding the gaze exists at this point.



fcv2

Fig. 5.2 Step-wise illustration for the threshold setting effect in order that adjusts Gabor transformation parameter based on the lighting conditions.

Tuning the threshold is a key to ensure accuracy of gaze tracking. For humans, sclera color is brighter than the surrounding region which contains iris, eyelids and pupil. Our technique incrementally update threshold from a state in which no edges is detected as shown in figure 5.2.a to a state in which gaze edge become visible as shown in figure 5.2.b. The

gaze edges represent the highest intensity change. This makes those edges appear first while filtering out most of the sclera region and the lower eyelid and the upper eyelid border as shown in figure 5.2.b. In such a specific area, leftmost non-zero point should be the right eye gaze while the right most non-zero point should be the left eye gaze. The group of points in which the intensity is changing forms the gaze edge. The threshold is fixed when such a state is achieved. Another threshold reconfiguration occurs in case the current settings start producing unwanted edges within the eye region. Validity checks are performed to avoid such situation and ensure output reliability. We tested our technique under different lighting conditions and eye states and it turns out to be of high robustness and accuracy. More results are presented in the experimental and discussion section.

Feedback and validity checks happen after each gaze point is found, each gaze pair coordinates are verified in terms of their alignment and the density of pixels between the two eyes. Left and right gazes pixel counts have to be similar. Differences between readings per gaze for adjacent frames shouldn't be bigger than the region of interest width. In the case of erroneous results due to an invalid parameter value, threshold resetting is applied.

Another issue that we tackled in our work is blink detection. The closed eye state can be classified from the open eye states together with their feature point's distribution. Straight forward method is to track the total number of feature points within a sequence of video frames. If the number is zero, which means the features are absent, then a blink has occurred.

5.3 Experiments

5.3.1 Evaluation data

We evaluated the accuracy of our method on challenging datasets called "Fifty People One Question"[120][121]. It is a TV show recorded under high environmental change with variety of races and facial alignments of subjects being interviewed. The dataset contains numerous videos with approximately 36,000 frames per each with a resolution of 1280×720 pixels.it contains variations in illumination, skin color and facial expression. We selected two videos within this data set, referred by the names: London and Brooklyn respectively. Samples are shown in Figure 5.3.



Fig. 5.3 The selected Dataset contains a variety of facial expressions, backgrounds and brightness levels, and in some cases, multiple faces in the same frame.

5.3.2 Evaluation Criteria

In order to assess the precision of gaze tracking we picked a common accuracy measure mentioned by Jesorsky et al. [122]. The relative error in eye detection is defined by

$$Error = \frac{\max(d_l, d_r)}{\|C_l - C_r\|}. \quad (5.9)$$

It depends on the distance between the true gaze positions C_l for the left gaze and C_r for the right gaze. It also depends on d_l and d_r , which refer to the detected location of the left and right gaze at certain frame. This metric will rely on image resolution which is the same in all our evaluation frames. Absolute distances are in pixels. We manually annotated all the gaze coordinates in pairs in all the testing frames due to the absence of such information.

If the error is less than 0.25, the result is deemed correct as the dominator of error equation is nearly twice the eye width [122]. To further analyze the results, we compared the detection rate when the error value gradually decreases from 0.25 to 0.05 in step size of 0.05. Detection accuracy refers to the ability of the system to identify the location of the gaze. The further the pupil coordinates are from the center of the pupil, the lower the detection accuracy.

5.3.3 State of the art algorithms

Our method: Gabor Filter gaze detector, which we refer to as GFD, as well as five other contemporaneous approaches for gaze detection are compared using the accuracy measure we mentioned in the previous section. The numerical experiments were executed on an I5 computer running 32 bit Windows 7 operating system with 2.1-GHz dual core processor, 2

GB of memory and a shared memory graphic card. The five algorithms that were included in the experiment are:

- (1) Automatic eye detection using "Intensity Filtering" and K-means clustering, referred to as KMC [123]. This is a less dynamic version of our approach. It uses static filter parameters and image enhancement that is applied to still images.
- (2) Iris recognition using a scalar-based template in Eigen-space, referred to as SBT [124]. It uses an enhanced template-matching technique involving an eigenspace transformation of the template item and of the image itself.
- (3) Eye recognition using the Eigeneyes method, referred to as EE [125]. It searches the image for possible eye candidates based on a dictionary of eyes transformed into the eigenspace.
- (4) Eye movement tracking system based on the camshift algorithm [126]. This technique proposes a real time tracker that requires the manually highlighting the pupil area. We refer to this as CS.
- (5) Haar detectors for face and facial feature detection [127]. It utilizes recent trained classifiers to track multiple facial features. We refer to this as CC.

5.4 Discussing Results

In figure 5.4, We show the gaze detection results using our technique for normal lighting conditions. The images at top row shows accurate detection while the images at the bottom shows mistaken detection. while In figure 5.5, We show the gaze detection results using our technique for dimmer lighting conditions. again, the images at top row shows accurate detection while the images at the bottom shows mistaken detection. Comparison results for our technique in addition to the other algorithms for Brooklyn and London videos are shown in tables 1 and 2.

Its worth mentioning that an inherent error can't be avoided in gaze direction estimation. This is because the human visual attention area covers a certain visual field without observing everything. Our technique is not able to detect gaze in eyes occlusion by hair or sun glasses. This effects our overall accuracy yet our technique still shows better performance compared to the state of the art algorithms.

Analysis of the detection process shows that most of the incorrect detections by the other methods are caused by lighting changes. More specifically, the method of intensity filtering



Fig. 5.4 Top row pictures illustrate the Gabor filtering technique ability to correctly detect gazes in normal lighting conditions. The gaze center is highlighted with a red cross to illustrate its position. The bottom row pictures show Gabor filtering technique mistaken detection results. The red cross are found near the face region but not accurately defining the gaze position.

Table 1 Accuracy and error ratios for six algorithms applied to the Brooklyn dataset; GFD = Gabor filter gaze detector, KMC = K-means clustering, SBT = scalar-based template, EE = Eigeneyes, CS = camshift classifier, CC = cascade classifier.

Error	Brooklyn DS					
	GFD	KMC	SBT	EE	CS	CC
0.05	79%	71%	51%	54%	63%	35%
0.1	83%	75%	58%	62%	69%	40%
0.15	86%	81%	63%	68%	72%	43%
0.2	89%	82%	75%	71%	75%	44%
0.25	95%	83%	83%	84%	89%	48%

(KMC) mainly uses specific parameter values for both the wavelet and brightness. If there is a change in light during the processing time, their method is unable to obtain satisfactory results. The methods of Eigen space transformation (SBT and EE) uses template matching information to extract eye segments. When the brightness changes, those method cannot efficiently obtain eye segments. The template matching methods (CS and CC) count mainly on an initial input of the pupil. Through the gradual change of brightness, the readings will be affected slightly. On the long run accuracy will deteriorate drastically.

Based on our experiments, our technique outperforms the other algorithms when it comes to accuracy. As previously mentioned, the Gabor transformation to an image is immune to salient changes in lighting which adds robustness under illumination change and user movement. Out of 36,000 frames our method could not detect eyes in 1,395 frames. This is due to eyes being closed, the subject moving faster than the frame rate or to the reflection of

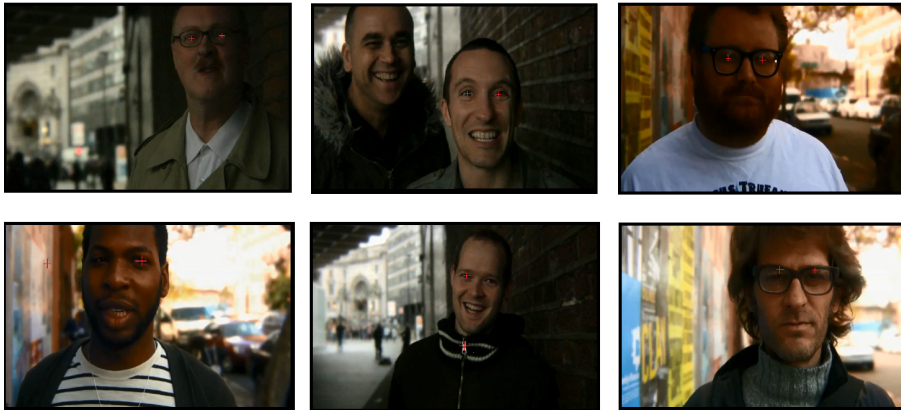


Fig. 5.5 Due to the lighting conditions variation the threshold gets automatically adjusted to detect gaze edge. The top row pictures show an accurate detection while the lower row shows inaccurate gaze detection. The gaze center is highlighted with a red cross to illustrate its position.

Table 2 Accuracy and error ratios for six algorithms applied to the London dataset; GFD = Gabor Filter gaze detector, KMC = K-means clustering, SBT = scalar-based template, EE = Eigeneyes, CS = camshift classifier, CC = cascade classifier.

Error	London DS					
	GFD	KMC	SBT	EE	CS	CC
0.05	65%	53%	46%	49%	51%	32%
0.1	71%	54%	52%	51%	62%	38%
0.15	73%	61%	58%	59%	66%	41%
0.2	79%	68%	68%	59%	71%	48%
0.25	85%	75%	71%	63%	72%	51%

eyeglasses, which changes the intensity values of pixels around the eyes. Figure 5.6 and 5.7 show that our method outperforms the other methods.

5.5 Conclusion

In this chapter, we introduce a gaze tracking technique by Gabor transform. Our method performance has been evaluated under various eye states and illumination conditions. After extracting the region of interest that contains the eyes, a Gabor filter gaze detector algorithm has been applied. This algorithm is able to precisely track gaze center in different light conditions. Furthermore, due to the nature of this transformation, our technique is able to solve the eye blink detection issue. Our solution based on the presence or absence of gaze edge is experimented to be accurate. We evaluated our technique and showed an accuracy of

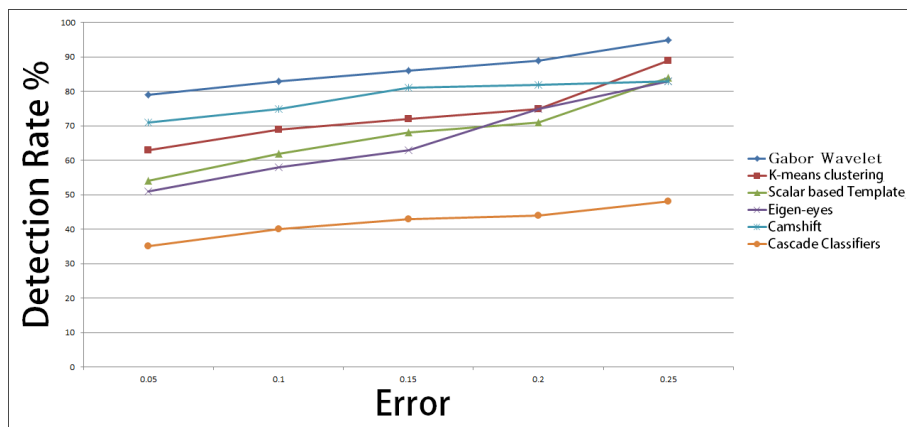


Fig. 5.6 Gaze Detection rate against error ranges. Brooklyn video comes with standard lighting conditions. Our technique shows higher detection rate compared to other techniques across all the error ranges.

85% and 95% in error level of 0.25 for dim and normal lighting conditions, respectively. Our work serves reliably for remote controlled system which allows standard head movement and alignment change. We also constructed an annotation set to refer to it when discussing accuracy and robustness that is available for other researchers. Our annotation includes frame by frame coordinates of eye pairs. Researchers can utilize these details for verifying the potential of new algorithms.

It is possible to utilize neural networks to track gazes by preparing applying Gabor filter to the network input. It can enhance neural network functionality by adding ability to compensate the loss of gaze tracking due to occlusion by predicting where the user is looking through other facial features.

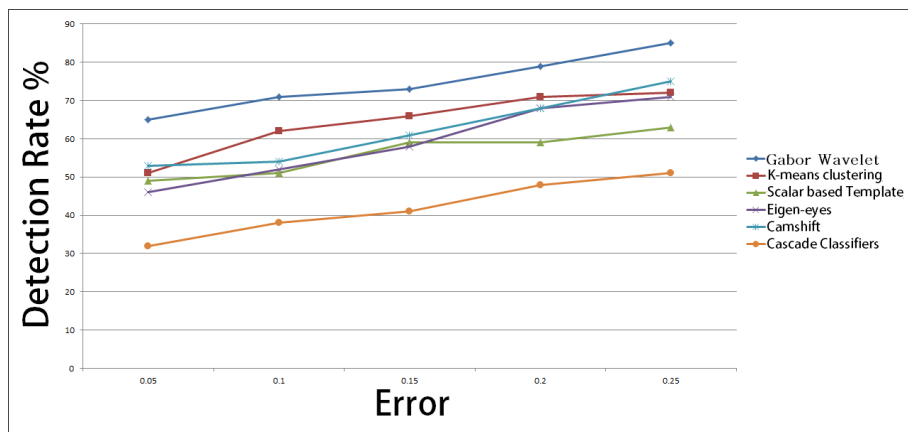


Fig. 5.7 Gaze Detection rate against error ranges. London video comes with dimmer lighting conditions and more challenging environment to detect the gaze edge. The Gabor filter detection rate is significantly less in dim lighting conditions compared to the rate in standard lighting conditions. Our technique shows higher detection rate compared to other techniques across all the error ranges.

Chapter 6

Conclusion

This work analyzes feature extraction and understanding in neural networks and their importance for emotion analysis. It discusses methods of Generalizing learned features in artificial systems to enhance sparsity and models localization abilities; and also extends localization abilities into temporal domain. This work discusses those aspects for emotion analysis resulting in improvements in emotion classification abilities and intensity estimation. The work starts with an overview of the relevant theoretical background in the field of neural network features understanding and learning process (chapter 2). The two aspects used most often in the literature (sparsity, generalization) are discussed and analyzed as to their suitability visual understanding. It worked out that both measures can be applied to understand the evolution of feature learning for affective computing. Interestingly it has been found that injecting noise at specific locations in a neural network model is able to obtain a better performance for both deep and shallow networks. The proposed method was demonstrated through classification problem of digits under variety of hand writing. The experimental findings match our hypothesis which is supported by mathematical evidence.

One challenge in artificial learning is the treatment of uncertain cases that the model encounters. In our case, this makes the description of an affective state subject to subjective interpretations, which depend not only on facial expressions but also on the various backgrounds surrounding the affective region. The many studies in correlating facial expressions to affective states focus on filtering salient and dominant features from other less meaningful features. Further problems with categorical descriptions are the many mutual features across different categories that lead to inaccurate emotional estimation. Therefore, in this work it is suggested utilizing noise as regularizer for its ability to suppress the learning process when the model is uncertain and increase the learning rate when encountering certain samples.

In chapter 2 it is also found that the internal representations of our model emerge upon introducing noise as regularizer. This could be attributed to network enhanced generalization

abilities. This is intensively utilized to enhance localization and sparsity of a deep neural network model of stacked Inception models in chapter 3. We suppress the number of effective weights by noise injection as a regularizer. Injecting noise directs the learning algorithm towards a solution with the least possible amount of zero weights. The noise was injected selectively to multiple joints within a deep CNN and its relationship with feature localization and performance was constructive. This is conducted through emotion classification experiments based on facial expressions. Salient improvement can be seen through the model activation maps and feature maps of the internal representation. Besides that, our experiments steadily result in accuracy improvements and clearer localization abilities of important features upon injecting Gaussian noise into selected joints of the neural network to classify the 7-emotional states in a variety of datasets.

Until this part of our work we consider an expression and its arousal as separate, seemingly independent components of an emotion. This means the basic principle of the temporal understanding was neglected. The expression and its arousal only together describe the unique emotional state. Therefore, our work is extended to analyze facial expression intensity estimation against the basic emotion states in chapter 4. We also analyzed the evolution of the emotion in terms of the emotional features locality and detected Micro-Expressions. Our model receives sequences that contain emotional transitions and estimates their intensity level. Our intensity estimation performance excels in terms accuracy, generalization for the hidden unit's activation and feature localization when compared to other intensity estimators. This can be observed through experiments and visualizations for the hidden layer representation. Those findings will help understanding emotional transitions by looking at the evolution the action units that define an emotion.

Based on the theoretical considerations of chapter 4, Emotional transition via facial expression is used to evaluate an affective intensity. It avoids some of the drawbacks of standard affection models by introducing temporal information. The data organization preserves the nature of an emotional arousal, avoids the dependency on labeled emotional state, and considers the facial expressions and arousal values as unique pairs describing the emotional state of a person. The developed approach has been evaluated in an experiment by variety of datasets and proved to be applicable and very suitable.

As identified in chapters 2, 3 and 4, neural networks inspired by biological brain structures have robust abilities to cope with recent research objectives. Our brain builds representations of the obtained information in a systematic way achieved by making small tweaks to an existing representation – its configuration contains significant information before any learning is conducted. The strengths of connections between neurons, or weights, do not start as random, nor does the structure of the connections. The fact that there is an initial

representation that works well for many tasks is supported by research, which suggests that as young as one-month old newborns can recognize faces demonstrated by their learning to differentiate between strangers and their parents. The more people we encounter the more need to make minor alterations to the neural networks residing in their brains. Similarly, neural networks manage to alter their internal weights to find optimism representation that serves the purpose of the model. Once they encounter a related task, both a human and neural network can use their previous representations of the problems to craft responses to new stimuli that they have not previously been exposed to in the learning process.

The more the human is exposed to similar problem the better the learning gets. The more problems that we encounter the better we become in tackling new problems, because relevant neuronal connections in the child's brain become more defined. In a very similar manner, a neural network that is exposed to the wide distribution of possible stimuli for the task in question, is better in responding to new stimuli from the same distribution that it was not previously exposed to. The regularization used to enhance the neural network learning helps structuring the knowledge that is obtained while learning.

We utilized the proposed architectures to extract features allowing interpreting emotions in a way similar to how humans sense an emotion. The concepts include robust understanding of emotions, transitional states and ability to detect micro expressions which is key to understand feelings at a very early stage of an emotional transition which is found vital in today's research needs.

We also proposed another measure of understanding emotions through the eyes. Our method is able to overcome a major challenge in common eye tracking that is offering readings in the wild. To verify the applicability of the concept two datasets were used. The evaluation results have shown that the concept is very suitable and applicable to different scenarios. This model can be used intensively for understanding emotions by linking certain eye movement during an interaction with certain feelings. We believe this extension is important for many emotion interpretation systems.

The robust feature extraction abilities of the proposed architectures prove the general applicability of the solutions developed in this thesis. Furthermore, it shows that with the solutions provided in this work has ability to generalize in many tasks from sensing via feature extraction, classification and emotion representation to finally using the emotion interpretation to estimate affective intensity.

Concluding, it can be said that the work described here analyzes several issues that have so far hindered the feature extraction and understanding. With the approaches suggested in this thesis it should be possible for HCI researchers and system developers alike to describe a person's affective state and to unambiguously assign engagement level for an emotion among

those states. We also implemented an eye tracking technique that will help future researchers sensing emotions and gather emotion-related data outside the lab, opening new possibilities to study human emotions in more realistic settings.

References

- [1] Picard, R. Affective computing. Technical Report 321, MIT Media Lab, Perceptual Computing Group, 1995.
- [2] Yoshua Bengio. Deep Learning of Representations: Looking Forward. arXiv.org, May 2013.
- [3] Zifeng Lian, Xiaojun Jing, Xiaohan Wang, Hai Huang, Youheng Tan, Yuanhao Cui, DropConnect Regularization Method with Sparsity Constraint for Neural Networks, Chinese Journal of Electronics, Volume: 25, Issue: 1 Pages 152-158, (2016).
- [4] Binbin Cao; Jianmin Li; Bo Zhang, Regularizing neural networks with adaptive local drop, International Joint Conference on Neural Networks, Pages 1-5, (2015).
- [5] Evgeny A. Smirnov, Denis M. Timoshenko, Serge N. Andrianov. Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks. AASRI Conference on Computational Intelligence and Bioinformatics, Pages 89-94, (2014).
- [6] Nishtha Tripathi, Avani Jadeja. A Survey of Regularization Methods for Deep Neural Network. International Journal of Computer Science and Mobile Computing, IJCSMC, Vol. 3, Issue. 11, Pages 429-436, (2014).
- [7] SH Inayoshi and T Kurita.: Improved Generalization by Adding both Auto-Association and Hidden-Layer- Noise to Neural-Network-Based-Classifiers. Machine Learning for Signal Processing, 2005 IEEE Workshop, Pages 141-146 (2005).
- [8] B. Poole, J. Sohl-Dickstein, and S. Ganguli. Analyzing noise in AutoEncoders and deep networks. arXiv:1406.1831 [cs].arXiv: 1406.1831. URL: <http://arxiv.org/abs/1406.1831>, June (2014).
- [9] Chi Sing Leung; Hong-Jiang Wang; John Sum, On the Selection of Weight Decay Parameter for Faulty Networks, IEEE Transactions on Neural Networks, Volume: 21, Issue: 8, Pages 1232-1244, (2010).

-
- [10] Takio Kurita, Hideki Asoh, Shinji Umeyama, Shotaro Akaho and Akitaka Hosomi. A Structural Learning by Adding Independent noises to Hidden Units. Proceedings of the IEEE International Conference on Neural Networks , IEEE World congress on Computational intelligence, Volume 1, Pages 275-278, (1994).
- [11] Kartik Audhkhasi, Osonde Osoba, Bart Kosko. Noise-enhanced convolutional neural networks, Neural Networks, Volume 78, Pages 15-23, (2016).
- [12] B. Poole, J. Sohl-Dickstein, and S. Ganguli. Analyzing noise in autoencoders and deep networks. arXiv:1406.1831 [cs]. arXiv: 1406.1831. URL: <http://arxiv.org/abs/1406.1831>, June (2014).
- [13] LeCun, Yann, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits." (1998).
- [14] D. S. Yeung, J. C. Li, W. W. Y. Ng and P. P. K. Chan, MLPNN Training via a Multiobjective Optimization of Training Error and Stochastic Sensitivity, in IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 5, Pages 978-992, (2016).
- [15] T. Kurita, T. Takahashi Viewpoint independent face recognition by competition of viewpoint dependent classifiers Neurocomputing, Pages 181-195, (2003).
- [16] P Vincent, H Larochelle, Y Bengio, P-A Manzagol, Extracting and composing robust features with denoising AutoEncoders, in Proc. of the 25th International Conference on Machine Learning (ICML),, Pages 1096-1103, (2008).
- [17] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach and James Martens. AAdding Gradient Noise Improves Learning for Very Deep Networks. arXiv:1511.06807. URL: <https://arxiv.org/abs/1511.06807>, November (2015).
- [18] D. Bolle, J. Busquets Blanco and T. Verbeirens. Multiplicative versus additive noise in multi-state neural networks. arXiv:cond-mat/0403020. URL: <https://arxiv.org/abs/cond-mat/0403020>, March (2004).
- [19] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. (2012).

- [20] Richard M. Zur, Yulei Jiang, Lorenzo L. Pesce, Karen Drukker. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Med Phys.* Pages 4810-4818. Published online (2009).
- [21] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, Hermann Ney, A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition, arXiv:1606.06871v1, (2015).
- [22] Guozhong An.:The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, Pages 643-674, (1996).
- [23] Takio Kurita, Hideki Asoh, Shinji Umeyama and Shotaro Akaho. Influence of Noises Added to Hidden Units on Learning of Multilayer Perceptrons and Structurization of Networks. *Systems and Computers in Japan*, Pages 257-266, Vol. 27, No. 11, (1996).
- [24] T.Kurita, H.Asho, S.Umeyama, S.Akaho, and A.Honomi, structural learning for multi layered perceptrons by adding independent noises," *IEICE Tech. Rep.*, Vol.93, No.124, (1993).
- [25] Kingma, D. P., and Ba, J. L. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, Pages: 1-13,(2015).
- [26] L. van der Maaten and G. E. Hinton, Visualizing data using t-Sne, *Journal of Machine Learning Research*, pp. 2579-2605, vol. 9, November (2008).
- [27] Ekman P, Friesen WV, Hager JC. *Facial Action Coding System*. Salt Lake City: Research Nexus; 2002.
- [28] B. V. Kumar, "Face expression recognition and analysis: the state of the art," Course Paper, *Visual Interfaces to Computer*, 2009.
- [29] E. Kim and S. Vangala, Deep Action Unit classification using a binned intensity loss and semantic context model, *23rd International Conference on Pattern Recognition (ICPR)*, page 4136-4141, 2016.
- [30] Pooya Khorrami Tom Le Paine Thomas S. Huang, Do Deep Neural Networks Learn Facial Action Units When Doing Expression Recognition, *International Conference on Computer Vision Workshop*, pages 19-27, 2015.
- [31] Sayan Ghosh, Eugene Laksana, Stefan Scherer, Louis-Philippe Morency, A Multi-label Convolutional Neural Network Approach to Cross-Domain Action Unit Detection, In *Proceedings of ACII 2015*, pages 19-27, 2015.

-
- [32] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. CVPR'16 (arXiv:1512.04150, 2015).
- [33] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Computer Vision-ECCV, pages 818-833. Springer, 2014.
- [34] C. Szegedy et al., Going deeper with convolutions, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1-9, 2015.
- [35] LSVRC. Results of the lsvrc challenge. [Online].Available: <http://www.image-net.org/challenges/LSVRC/2014/results> 2014.
- [36] Chi Sing Leung; Hong-Jiang Wang; John Sum, On the Selection of Weight Decay Parameter for Faulty Networks, IEEE Transactions on Neural Networks, Volume: 21, Issue: 8, Pages 1232-1244, 2010.
- [37] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567, 2015.
- [38] Guozhong An.: The effects of adding noise during backpropagation training on a generalization performance. Neural computation, Pages 643-674, (1996).
- [39] C.-D. Caeanu, "Face expression recognition: A brief overview of the last decade," in Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on. IEEE, Pages 157-161, 2013.
- [40] V. Bettadapura, "Face expression recognition and analysis: the state of the art," arXiv preprint:1203.6722, 2012.
- [41] Yoshua Bengio. Deep Learning of Representations: Looking Forward. arXiv.org, May 2013.
- [42] Zifeng Lian, Xiaojun Jing, Xiaohan Wang, Hai Huang, Youheng Tan, Yuanhao Cui, DropConnect Regularization Method with Sparsity Constraint for Neural Networks, Chinese Journal of Electronics, Volume: 25, Issue: 1 Pages 152-158, 2016.
- [43] Motaz Sabri, Takio Kurita. Effect of Additive Noise for Multi-Layered Perceptron with AutoEncoders. IEICE TRANSACTIONS on Information and Systems 100.7, Pages 1494-1504, 2017.

- [44] Takio Kurita, Hideki Asoh, Shinji Umeyama, Shotaro Akaho and Akitaka Hosomi. A Structural Learning by Adding Independent noises to Hidden Units. Proceedings of the IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence, Volume 1, Pages 275-278, 1994.
- [45] Kartik Audhkhasi, Osonde Osoba, Bart Kosko. Noise-enhanced convolutional neural networks, *Neural Networks*, Volume 78, Pages 15-23, 2016.
- [46] Lundqvist, D., Flykt, A., and Öhman, A. (1998). *The Karolinska Directed Emotional Faces - KDEF*, CD-ROM from Department of Clinical Neuroscience, Psychology Section, Karolinska Institutet, ISBN 91-630-7164-9.
- [47] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. The Extended Cohn-Kande Dataset (CK+): A complete facial expression dataset for action unit and emotion-specified expression. *CVPR*, 2010.
- [48] Ruiz-Garcia, A., Elshaw, M., Altahhan, A., and Palade, V. Emotion Recognition Using Facial Expression Images for a Robotic Companion. In *International Conference on Engineering Applications of Neural Networks* (pp. 79-93).
- [49] YY. Sun, L. Ding, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. *CoRR*, abs/1502.00873, 2015.
- [50] Florian Schroff, Dmitry Kalenichenko, James Philbin, FaceNet: A Unified Embedding for Face Recognition and Clustering, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815-823, 2015.
- [51] Gil Levi , Tal Hassner, Age and Gender Classification Using Convolutional Neural Networks, *IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG)*, at the *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Boston, June 2015.
- [52] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.
- [53] Richard M. Zur, Yulei Jiang, Lorenzo L. Pesce, Karen Drukker. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Med Phys*.Pages 4810-4818. Published online 2009.

- [54] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, Hermann Ney, A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition, arXiv:1606.06871v1, 2015.
- [55] I. Song, H.-J. Kim, and P. B. Jeon, “Deep learning for real-time robust facial expression recognition on a smartphone,” in Consumer Electronics (ICCE), 2014 IEEE International Conference on IEEE, Pages 564-567, 2014.
- [56] Kanade, T., Cohn, J. F., and Tian, Y. Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG’00), Grenoble, France, Pages 46-53, 2000.
- [57] Alessandro E.P. Villa, Paolo Masulli, Antonio J. Pons Rivero. Deep Learning for Emotion Recognition in Faces. 25th International Conference on Artificial Neural Networks, Pages 38-47, 2016.
- [58] Kukla, E., and Nowak, P. Facial Emotion Recognition Based on Cascade of Neural Networks. Advances in Intelligent Systems and Computing, 67-78, 2015.
- [59] van Kuilenburg, H., Wiering, M., den Uyl, M. A model based method for facial expression recognition. Lectures Notes in Computer Science: Vol. 3720. Machine Learning: ECML, Berlin, Germany: Springer-Verlag pages 194-205, 2005.
- [60] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.
- [61] S. Kaltwang, O. Rudovic, and M. Pantic. Continuous pain intensity estimation from facial expressions. In Advances in Visual Computing, pages 368–377. Springer, 2012.
- [62] Z. Ambadar, J. Schooler, and J. F. Cohn. Deciphering the enigmatic face The importance of facial dynamics in interpreting subtle facial expression. Psychological Science, 2005.
- [63] Yang, Peng and Liu, Qingshan and Metaxas, Dimitris. RankBoost with $l(1)$ regularization for Facial Expression Recognition and Intensity Estimation. Proceedings of the IEEE International Conference on Computer Vision. pages 1018–1025, 2009.
- [64] Rui Zhao, Quan Gan, Shangfei Wang, and Qiang Ji, Facial expression intensity estimation using ordinal information, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3466—3474, 2016.

- [65] Nicholson, J. and Takahashi, K. and Nakatsu, R., Emotion Recognition in Speech Using Neural Networks, *Neural Computing and Applications*, pages 290–296, 2000.
- [66] van Kuilenburg, H., Wiering, M., den Uyl, M.: A model-based method for automatic facial expression recognition. In: Gama, J., et al. (eds.) *ECML 2005. LNCS (LNAI)*, vol. 3720, pages 194–205. Springer, 2005.
- [67] Battiti, R.: Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Network*, pages 537–550, 1994.
- [68] M. S. Bartlett J. C. Hager P. Ekman T. J. Sejnowski Measuring facial expressions by computer image analysis. *Psychophysiology*, pages 253–264, 1999.
- [69] P. Ekman and W. V. Friesen. *Manual for the facial action coding system*. Consulting Psychologists Press, 1978.
- [70] M. Batty, M.J. Taylor Early processing of the six basic facial emotional expressions *Brain Res. Cogn. Brain Res.*, pages 613–620, 2003
- [71] H. Kobayashi, F. Hara, Recognition of Six Basic Facial Expressions and Their Strength by Neural Network, *Proc. Int’l Workshop Robot and Human Comm.*, pages 381–386, 1992.
- [72] Y. Li, S. M. Mavadati, M. H. Mahoor, and Q. Ji. A unified probabilistic framework for measuring the intensity of spontaneous facial action units. In *FG*, pages 1–7, 2013.
- [73] O. Rudovic, V. Pavlovic, and M. Pantic. Context-sensitive dynamic ordinal regression for intensity estimation of facial action units. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 944–958, 2015.
- [74] A. Savran, B. Sankur, and M. T. Bilge. Regression-based intensity estimation of facial action units. *Image and Vision Computing*, pages 774–784, 2012.
- [75] Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: *ICML Workshops 2015*
- [76] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, L. Jackel, Y. L. Cun, C. Moore, E. Sackinger, R. Shah, "Signature verification using a 'Siamese' time delay neural network", *Int. J. Pattern Recognit. Artif. Intell. (IJPRAI)*, pages 669–688, 1993.

- [77] Lorenzo Baraldi , Costantino Grana , Rita Cucchiara, A Deep Siamese Network for Scene Detection in Broadcast Videos, Proceedings of the 23rd ACM international conference on Multimedia, pages 26–30, 2015.
- [78] Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: Feragen, A., Pelillo, M., Loog, M. (eds.), pages 84—92, 2015.
- [79] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. ICCV, 2015.
- [80] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. CVPR’16
- [81] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Computer Vision-ECCV, pages 818–833, 2014.
- [82] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. The Extended Cohn-Kande Dataset (CK+): A complete facial expression dataset for action unit and emotion-specified expression. Paper presented at the Third IEEE Workshop on CVPR for Human Communicative Behavior Analysis,CVPR4HB, 2010.
- [83] J. Reilly, J. Ghent, and J. McDonald. Non-linear approaches for the classification of facial expressions at varying degrees of intensity. IMVIP07, pages 125—132, 2007.
- [84] K. K. Lee and Y. Xu. Real-time estimation of facial expression intensity. International Conference on Robotics and Automation, 2003.
- [85] G. Littlewort, M. S. Bartlett, I. Fasel, J. Susskind, and J. Movellan. Dynamics of facial expression extracted automatically from video. J. Image and Vision Computing, 2006.
- [86] S. Koelstra and M. Pantic. Non-rigid registration using free-form deformations for recognition of facial actions and their temporal dynamics. Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (FG’08), Amsterdam, 2008.
- [87] S. Kaltwang, O. Rudovic, M. Pantic, Continuous pain intensity estimation from facial expressions, Proc. Int. Symp. Visual Comput., pages 368–377, 2012.
- [88] P. Baldi and Y. Chauvin, “Neural networks for fingerprint recognition,” Neural Computation, vol. 5, no. 3, pages 402–418, 1993.

- [89] S. Chopra, R. Hadsell, and Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in CVPR (1), pages 539–54, 2005.
- [90] Dong Yi, Zhen Lei, and Stan Z. Li. Deep metric learning for practical person reidentification. CoRR, abs/1407.4979, 2014.
- [91] Zheng, L., Duffner, S., Idrissi, K. et al. Siamese multi-layer perceptrons for dimensionality reduction and face identification. *Multimedia Tools and Applications*, Volume 75, Issue 9, pages 5055–5073, 2016.
- [92] Vijay Kumar, B.G., Carneiro, G., Reid, I.: Learning local image descriptors with deep Siamese and triplet convolutional networks by minimising global loss functions. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [93] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. CoRR, abs/1404.4661, 2014.
- [94] R. Jin, H. Valizadegan, and H. Li. Ranking refinement and its application to information retrieval. *International conference on World Wide Web*, pages 397–406, 2008.
- [95] Freund, Y., Iyer, R., Schapire, R.E., and Singer, Y. (1998). An efficient boosting algorithm for combining preferences. *Machine Learning: Proceedings of the Fifteenth International Conference*.
- [96] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems*, pages 115–132, 1999.
- [97] N. Aifanti, C. Papachristou and A. Delopoulos, "The MUG Facial Expression Database," in Proc. 11th Int. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), Desenzano, Italy, April 12-14 2010.
- [98] Motaz Sabri, Takio Kurita, Facial expression intensity estimation using Siamese and triplet networks, *Neurocomputing*, ISSN 0925-2312, 2018.
- [99] Hansen DW, Ji Q. In the eye of the beholder: a survey of models for eyes and gaze. *Pattern Anal Mach Intell IEEE Transac* 32(3), pp.478–500, (2010).
- [100] S. V. Sheela and Abhinand P, "Iris detection for gaze tracking using video frames," *Advance Computing Conference (IACC)*, 2015 IEEE International, Bangalore, pp.629-633, 2015.

- [101] Y. m. Cheung and Q. Peng, "Eye Gaze Tracking With a Web Camera in a Desktop Environment," in *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 4, pp. 419-430, Aug. 2015.
- [102] Li, D., Winfield, D., Parkhurst, D. J. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Computer Vision and Pattern Recognition-Workshops. CVPRWorkshops. IEEE Computer Society Conference*, pp.79–79, (2005).
- [103] Timm, F., Barth, E. Accurate Eye Centre Localisation by Means of Gradients. In *VISAPP*, pp.125–130, (2011).
- [104] Li, X., Sheng, B., Wu, W., Ma, L., and Li, P. . Accurate gaze tracking from single camera using Gabor corner detector. *Multimedia Tools and Applications*, 75(1), pp.221-239. (2016)
- [105] Guestrin ED, Eizenman E General theory of remote gaze estimation using the pupil center and corneal reflections. *Biomed Eng IEEE Transac* 53(6), pp.1124–1133,(2006).
- [106] Zhu Z, Ji Q Novel eye gaze tracking techniques under natural head movement. *Biomed Eng IEEE Transac* 54(12), pp.2246–2260, (2007).
- [107] Villanueva A, Cabeza R A novel gaze estimation system with one calibration point. *Syst Man Cybern Part B: Cybern IEEE Transac* 38(4):1123–1138 *Multimed 236 Tools Appl* 75, pp.221–239, (2008).
- [108] Valenti R, Sebe N, Gevers T Combining head pose and eye location information for gaze estimation. *Image Process IEEE Transac* 21(2):, pp.802–815, (2012).
- [109] Hennessey, C., Nouredin, B., Lawrence, P. A single camera eye-gaze tracking system with free head motion. In *Proceedings of the 2006 symposium on Eye tracking research and applications*, pp.87–94, (2006).
- [110] Coutinho, F. L., Morimoto, C. H . Free head motion eye gaze tracking using a single camera and multiple light sources. In *Computer Graphics and Image Processing, 2006. SIBGRAPI'06. 19th Brazilian Symposium*, pp.171–178, (2006).
- [111] Li, D., Babcock, J., Parkhurst, D. J. openEyes: a low-cost head-mounted eye-tracking solution. In *Proceedings of the 2006 symposium on Eye tracking research applications*, pp.95–100, (2006).

- [112] Holland, C., Garza, A., Kurtova, E., Cruz, J., and Komogortsev, O. Usability evaluation of eye tracking on an unmodified common tablet. In CHI Extended Abstracts (2013)
- [113] Wood, E., and Bulling, A. EyeTab: Model-based gaze estimation on unmodified tablet computers. In Symposium on Eye-Tracking Research and Applications (2014).
- [114] Kumar, N., Kohlbecher, S., Schneider, E. A novel approach to video-based pupil tracking. In Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference, pp.1255–1262, (2009).
- [115] Shi, J., Tomasi, C. Good features to track. In Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 I.E. Computer Society Conference, pp.593–600, (1994).
- [116] Harris C, Stephens M. A combined corner and edge detector. Proc 4th Alvey Vision Conf, 15:50, (1988).
- [117] Gao X, Sattar F, Venkateswarlu R Multiscale corner detection of gray level images based on gabor wavelet transform. Circ Syst Video Technol IEEE Transac 17(7), pp.868–875, (2007).
- [118] S. Liao, A. Jain, and S. Li. A fast and accurate unconstrained face detector. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.
- [119] Dun Mao, YueYun Cao, JiangHu Xu and Ke Li, "Object tracking integrating template matching and mean shift algorithm," Multimedia Technology (ICMT), 2011 International Conference on, Hangzhou, pp.3583-3586, 2011.
- [120] Fifty People One Question. (2009, Jan. 30). Fifty People, One Question: Brooklyn. Available: <https://www.youtube.com/watch?v=VJAUGg4081Q>. Accessed Oct. 24, 2016.
- [121] Fifty People, One Question. (2009, Jan. 31). Fifty People, One Question: London. Available: <https://www.youtube.com/watch?v=aQk30nYUOAw>. Accessed Oct. 24, 2016.
- [122] Jesorsky O, Kirchberg K.J, Frischholz R.W. Robust face detection using the hausdorff distance. AVBPA 2091:90-95. doi: 10.1016/j.patcog.2006.04.019, (2001).
- [123] Zhiming Qian, Dan Xu. Automatic eye detection using intensity Filtering and K-means clustering. Pattern Recognition Letters 31:1633-1640. doi: 10.1016/j.patrec.2010.05.012, (2010).

- [124] Abhijit Das, Ranjan Parekh. Iris Recognition using a Scalar based Template in Eigenspace. *International Journal of Computer Science and Telecommunications* 52: pp.49-74. doi: 10.5120/8314-1247,(2012).
- [125] Khamael Abbas AL-phalahi Al-dualim, Eye Recognition Technique Based on Eigeneyes Method. *International Conference on Software and Computer Applications* 9:212-219. doi:TP391.41,(2011).
- [126] Huang CR, Lv XQ, Zhao J, Ren QS, Chai XY. Implementation of eye movement tracking system based on camshift algorithm. *Chinese journal of medical instrumentation* 4: pp.239-342. PMID:19938517, (2009).
- [127] Castrillon-santana M, Dwniz-suarez O, Anton-canalís L, Lorenzo-navarro J. Face and facial feature detection evaluation performance evaluation of public domain HAAR detectors for face and facial feature detection. *Third Int. conf. in computer vision theory and applications* 52 pp.167-172. doi: 10.1.1.77.7484, (2008).
- [128] A. Savran, B. Sankur, and M. T. Bilge. Regression-based intensity estimation of facial action units. *Image and Vision Computing*, pages 774—784, (2012).
- [129] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, (2016).
- [130] Battiti, R. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Network*, pages 537–550, (1994).
- [131] Dong Yi, Zhen Lei, and Stan Z. Li. Deep metric learning for practical person reidentification. *CoRR*, abs/1407.4979, (2014).
- [132] Freund, Y., Iyer, R., Schapire, R.E., and Singer, Y. An efficient boosting algorithm for combining preferences. *Machine Learning: Proceedings of the Fifteenth International Conference*, (1998).
- [133] G. Littlewort, M. S. Bartlett, I. Fasel, J. Susskind, and J. Movellan. Dynamics of facial expression extracted automatically from video. *Image and Vision Computing*, (2006).
- [134] H. Kobayashi, F. Hara. Recognition of Six Basic Facial Expressions and Their Strength by Neural Network. *Proc. Int'l Workshop Robot and Human Comm.*, pages 381–386, (1992).
- [135] Hoffer, E., Ailon, N. Deep metric learning using triplet network. *Similarity-Based Pattern Recognition*, pages 84—92,(2015).

- [136] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, L. Jackel, Y. L. Cun, C. Moore, E. Sackinger, R. Shah. Signature verification using a 'Siamese' time delay neural network. *Int. J. Pattern Recognit. Artif. Intell. (IJPRAI)*, pages 669–688,(1993).
- [137] J. Reilly, J. Ghent, and J. McDonald. Non-linear approaches for the classification of facial expressions at varying degrees of intensity. *IMVIP07*, pages 125—132,(2007).
- [138] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. *CoRR*, abs/1404.4661,(2014).
- [139] K. K. Lee and Y. Xu. Real-time estimation of facial expression intensity. *International Conference on Robotics and Automation*,(2003).
- [140] Koch, G., Zemel, R., Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In: *ICML Workshops*,(2015).
- [141] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. The Extended Cohn-Kande Dataset (CK): A complete facial expression dataset for action unit and emotion-specified expression. *Third IEEE Workshop on CVPR for Human Communicative Behavior Analysis, CVPR4HB*,(2010).
- [142] Lorenzo Baraldi , Costantino Grana , Rita Cucchiara. A Deep Siamese Network for Scene Detection in Broadcast Videos. *Proceedings of the 23rd ACM international conference on Multimedia*, pages 26–30,(2015).
- [143] M. S. Bartlett J. C. Hager P. Ekman T. J. Sejnowski. Measuring facial expressions by computer image analysis. *Psychophysiology*, pages 253–264,(1999).
- [144] M. Batty, M.J. Taylor. Early processing of the six basic facial emotional expressions *Brain Res. Cogn. Brain Res.*, pages 613–620,(2003).
- [145] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision-ECCV*, pages 818–833,(2014).
- [146] M.F. Valstar, M. Pantic. . Induced Disgust, Happiness and Surprise: an Addition to the MMI Facial Expression Database. *Proceedings of the International Language Resources and Evaluation Conference, Malta*,(2010).
- [147] Motaz Sabri, Takio Kurita. Improvement of feature localization for facial expressions by adding noise. *Journal of Kansei Engineering International*, Accepted and to be published,(2018).

- [148] Nicholson, J. and Takahashi, K. and Nakatsu, R. Emotion Recognition in Speech Using Neural Networks. *Neural Computing and Applications*, pages 290–296, (2000).
- [149] N. Aifanti, C. Papachristou and A. Delopoulos. The MUG Facial Expression Database. 11th Int. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), 12–14, (2010).
- [150] O. Rudovic, V. Pavlovic, and M. Pantic. Context-sensitive dynamic ordinal regression for intensity estimation of facial action units. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 944–958, (2015).
- [151] P. Ekman and W. V. Friesen. (1978). *Manual for the facial action coding system*. Consulting Psychologists Press, (1978).
- [152] P. Baldi and Y. Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, vol. 5, no. 3, pages 402–418, (1993).
- [153] Qu F, Wang SJ, Yan WJ, Li H, Wu S, Fu X. CAS (ME)²: a database for spontaneous macro-expression and micro-expression spotting and recognition. *IEEE Trans Affect Comput*, (2017).
- [154] Q. Wu, X. Shen, and X. Fu. (2011). The machine knows what you are hiding: an automatic micro-expression recognition system. in *Affective Computing and Intelligent Interaction*. Springer. pages 152–162, (2011).
- [155] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems*, pages 115–132, (1999).
- [156] R. Jin, H. Valizadegan, and H. Li. Ranking refinement and its application to information retrieval. *International conference on World Wide Web*, pages 397–406, (2008).
- [157] S. Koelstra and M. Pantic. Non-rigid registration using free-form deformations for recognition of facial actions and their temporal dynamics. *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition* pages 1–8, (2008).
- [158] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *CVPR (1)*, pages 539–54, (2005).
- [159] S. Kaltwang, O. Rudovic, M. Pantic. Continuous pain intensity estimation from facial expressions. *Proc. Int. Symp. Visual Comput.*, pages 368–377, (2012).

- [160] S. Polikovsky, Y. Kameda, and Y. Ohta. Facial micro-expressions recognition using high speed camera and 3d-gradient descriptor. in *Crime Detection and Prevention (ICDP 2009)*, 3rd International Conference on. IET. pages 1–6,(2009).
- [161] V. Kuilenburg, H., Wiering, M., den Uyl, M. A model-based method for automatic facial expression recognition. In: Gama, J., et al. (eds.) *ECML 2005. LNCS (LNAI)*, vol. 3720, pages 194—205. Springer,(2005).
- [162] V. Kumar, B.G., Carneiro, G., Reid, I. Learning local image descriptors with deep Siamese and triplet convolutional networks by minimising global loss functions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,(2016).
- [163] W.-J. Yan, Q. Wu, Y.-J. Liu, S.-J. Wang, and X. Fu. Casme database:a dataset of spontaneous micro-expressions collected from neutralized faces. in *Automatic Face and Gesture Recognition, 10th International Conference and Workshops on IEEE*. pages 1–7,(2013).
- [164] W.-J. Yan, X. Li, S.-J. Wang, G. Zhao, Y.-J. Liu, Y.-H. Chen, and X. Fu. (2014). Casme ii: An improved spontaneous micro-expression database and the baseline evaluation. *PloS one*, vol. 9, no. 1, e86041,(2014).
- [165] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. *ICCV*,(2015).
- [166] Yang, Peng and Liu, Qingshan and Metaxas, Dimitris. RankBoost with $l(1)$ regularization for Facial Expression Recognition and Intensity Estimation. *Proceedings of the IEEE International Conference on Computer Vision*. pages 1018–1025,(2009).
- [167] Y. Li, S. M. Mavadati, M. H. Mahoor, and Q. Ji. A unified probabilistic framework for measuring the intensity of spontaneous facial action units. In *FG*, pages 1–7,(2013).
- [168] Z. Ambadar, J. Schooler, and J. F. Cohn. (2005). Deciphering the enigmatic face The importance of facial dynamics in interpreting subtle facial expression. *Psychological Science*,(2005).
- [169] Zheng, L., Duffner, S., Idrissi, K. et al. Siamese multi-layer perceptrons for dimensionality reduction and face identification. *Multimedia Tools and Applications*, Volume 75, Issue 9, pages 5055—5073,(2016).

- [170] Zhao Rui, Quan Gan, Shangfei Wang, and Qiang Ji. Facial expression intensity estimation using ordinal information. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3466—3474,(2016).

Appendix A

This appendix contains derivations of equations that were considered too lengthy or not crucial to the body of the paper.

A.0.1 Additive Noise Update rule for u_{hi} - Equation 2.18

$$\tilde{\ell}_1 = \frac{1}{2} \sum_{n=1}^N \sum_{j=1}^I (\tilde{z}_{nj} - z_{nj})^2 = \ell_1 + \sum_{n=1}^N \sum_{j=1}^I (\Delta z_{nj} \delta_{nj} + \frac{\Delta z_{nj}^2}{2}) \quad (\text{A1})$$

where $\delta_{nj} := z_{nj} - \tilde{z}_{nj}$. Thus

$$\frac{\partial \tilde{\ell}_1}{\partial u_{hi}} = \frac{\partial \ell_1}{\partial u_{hi}} + \frac{\partial}{\partial u_{hi}} \sum_{n=1}^N \sum_{j=1}^I (\delta_{nj} \Delta z_{nj} + \frac{\Delta z_{nj}^2}{2}) \quad (\text{A2})$$

For simplicity we approximate Δz_{nj}^2 to be $\Delta z_{nj}^2 \approx \sum_{h=1}^H (w_{jh} \omega_{nh} \epsilon_h)^2$ then we have

$$\frac{\partial \Delta z_{nj}^2}{\partial u_{hi}} = 4w_{jh}^2 \epsilon_h^2 \omega_{nh} v_{nh} x_{ni} \quad (\text{A3})$$

$$\begin{aligned} \frac{\partial \delta_{nj} \Delta z_{nj}}{\partial u_{hi}} = \\ \left(w_{jh} \Delta z_{nj} + ((1 - 2g_{nh}) \epsilon_h + \frac{1}{2} \chi_{nh} \epsilon_h^2) \delta_{nj} \right) \omega_{nh} x_{ni} \end{aligned} \quad (\text{A4})$$

A4 is obtained through chain rule. The partial derivative

$$\frac{\partial}{\partial u_{hi}} (\Delta z_{nj} \delta_{nj} + \frac{\Delta z_{nj}^2}{2}) =$$

$$\begin{aligned}
& w_{jh}\Delta z_{nj}\omega_{nh}x_{ni} + 2w_{jh}^2\epsilon_h^2\omega_{nh}v_{nh}x_{ni} \\
& + ((1 - 2g_{nh})\epsilon_h + \frac{1}{2}\chi_{nh}\epsilon_h^2)w_{jh}\delta_{nj}\omega_{nh}x_{ni}
\end{aligned} \tag{A5}$$

$$\frac{\partial \tilde{\ell}_2}{\partial u_{hi}} = \sum_{n=1}^N \sum_{k=1}^K (\tilde{y}_{nk} - t_{nk}) \left(\frac{\partial \rho_{nk}}{\partial u_{hi}} + \frac{\partial \Delta \rho_{nk}}{\partial u_{hi}} \right) \tag{A6}$$

$$\frac{\partial \Delta \rho_{nk}}{\partial u_{hi}} = 2v_{kh}v_{nh}x_{ni}\epsilon_h + \frac{1}{2}v_{kh}\chi_{nh}\omega_{nh}x_{ni}\epsilon_h^2 \tag{A7}$$

$$\frac{\partial \rho_{nk}}{\partial u_{hi}} = v_{kh}\omega_{nh}x_{ni} \tag{A8}$$

A8 is found through chain rule, we have

$$\begin{aligned}
& \frac{\partial \rho_{nk}}{\partial u_{hi}} + \frac{\partial \Delta \rho_{nk}}{\partial u_{hi}} = \\
& v_{kh}\omega_{nh}x_{ni} \left(1 + \frac{\chi_{nh}\epsilon_h^2}{2} \right) + 2v_{kh}v_{nh}x_{ni}\epsilon_h^2
\end{aligned} \tag{A9}$$

A5 and A9 together with the summations given in A1 and A6 form the equation 2.18. Equation 2.27 is derived similarly.

A.0.2 Expectation of Equation $\frac{\partial \tilde{\ell}_1}{\partial u_{hi}}$ - Equation 2.20

$$\begin{aligned}
\mathbb{E} \left[\frac{\partial \tilde{\ell}_1}{\partial u_{hi}} \right] &= \frac{\partial \ell_1}{\partial u_{hi}} + \sum_{n=1}^N \sum_{j=1}^I w_{jh}\omega_{nh}x_{ni}\sigma^2 \left(\sum_{h'=1}^H w_{jh'}v_{nh'} \right) \\
&+ \frac{\partial \ell_1}{\partial u_{hi}} \frac{\chi_{nh}\sigma^2}{2} + 2 \sum_{j=1}^I w_{jh}^2\omega_{nh}v_{nh}x_{ni}\sigma^2
\end{aligned} \tag{A10}$$

$$\begin{aligned}
&= \frac{\partial \ell_1}{\partial u_{hi}} \left(1 + \frac{\chi_{nh}\sigma^2}{2} \right) + 2 \sum_{n=1}^N \sum_{j=1}^I w_{jh}^2\omega_{nh}v_{nh}x_{ni}\sigma^2 \\
&+ \sum_{n=1}^N \sum_{j=1}^I w_{jh}\omega_{nh}x_{ni}\sigma^2 \left(\sum_{h'=1}^H w_{jh'}v_{nh'} \right)
\end{aligned} \tag{A11}$$

$$\begin{aligned}
&= \frac{\partial \ell_1}{\partial u_{hi}} \left(1 + \frac{\chi_{nh} \sigma^2}{2} \right) + 3 \sum_{n=1}^N \sum_{j=1}^I w_{jh}^2 \omega_{nh} \nu_{nh} x_{ni} \sigma^2 \\
&\quad + \sum_{n=1}^N \sum_{j=1}^I w_{jh} \omega_{nh} x_{ni} \sigma^2 \left(\sum_{h'=1, h' \neq h}^H w_{jh'} \nu_{nh'} \right)
\end{aligned} \tag{A12}$$

A.0.3 Expectation of Equation $\frac{\partial \tilde{\ell}_2}{\partial u_{hi}}$ - Equation 2.20

$$\mathbb{E} \left[\frac{\partial \ell_2}{\partial u_{hi}} \right] = \mathbb{E} \left[(\tilde{y}_{nk} - t_{nk}) \frac{\partial \rho_{nk}}{\partial u_{hi}} \left(1 + \frac{\chi_{nh} \epsilon_h^2}{2} \right) \right].$$

In order to find $\mathbb{E}[\tilde{y}_{nk} - t_{nk}]$, Jensen's inequality is used:

$$\mathbb{E}[\log \tilde{y}_{nk}] \leq \log \mathbb{E}[\tilde{y}_{nk}] \tag{A13}$$

$$\begin{aligned}
\mathbb{E}[\log \tilde{y}_{nk}] &= \mathbb{E}[\tilde{\rho}_{nk}] - \mathbb{E} \left[\log \sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}} \right] \\
&= \rho_{nk} + \mathbb{E}[\Delta \rho_{nk}] - \mathbb{E} \left[\log \sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}} \right] \\
&= \rho_{nk} + \left(\sum_{h=1}^H v_{kh} \nu_{nh} \right) \sigma^2 - \mathbb{E} \left[\log \sum_{n=1}^N \sum_{j=1}^K e^{\tilde{\rho}_{nj}} \right]
\end{aligned} \tag{A14}$$

By Jensen's inequality again leads to equations 2.21 and 2.22.

