Anomaly Detection in Video

by

Tran Thi Minh Hanh

Submitted in accordance with the requirements for the degree of Doctor of Philosophy

The University of Leeds

School of Computing

June 2018

Declarations

The candidate confirms that the work submitted is his/her own, except where work which has formed part of a jointly authored publication has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Some parts of the work presented in this thesis have been published in the following articles:

Hanh T. M. Tran and David C. Hogg. Anomaly Detection using a Convolutional Winner-Take-All Autoencoder. *Proceedings of the British Machine Vision Conference (BMVC), BMVA Press, September 2017.*

The candidate confirms that the above jointly-authored publications are primarily the work of the first author. The role of the second author was purely supervisory.

Abstract

Anomaly detection is an area of video analysis that has great importance in automated surveillance. Although it has been extensively studied, there has been little work on using deep convolutional neural networks to learn spatio-temporal feature representations. In this thesis we present novel approaches for learning motion features and modelling normal spatio-temporal dynamics for anomaly detection.

The contributions are divided into two main chapters. The first introduces a method that uses a convolutional autoencoder to learn motion features from foreground optical flow patches. The autoencoder is coupled with a spatial sparsity constraint, known as Winner-Take-All, to learn shift-invariant and generic flow-features. This method solves the problem of using hand-crafted feature representations in state of the art methods. Moreover, to capture variations in scale of the patterns of motion as an object moves in depth through the scene, we also divide the image plane into regions and learn a separate normality model in each region. We compare the methods with state of the art approaches on two datasets and demonstrate improved performance.

The second main chapter presents a end-to-end method that learns normal spatio-temporal dynamics from video volumes using a sequence-to-sequence encoder-decoder for prediction and reconstruction. This work is based on the intuition that the encoder-decoder learns to estimate normal sequences in a training set with low error, thus it estimates an abnormal sequence with high error. Error between the network's output and the target output is used to classify a video volume as normal or abnormal. In addition to the use of reconstruction error, we also use prediction error for anomaly detection.

We evaluate the second method on three datasets. The prediction models show comparable performance with state of the art methods. In comparison with the first proposed method, performance is improved in one dataset. Moreover, running time is significantly faster.

Acknowledgements

First of all I would like to express my utmost gratitude to my supervisor Prof. David Hogg. His guidance, feedback and encouragement have inspired me to pursue novel ideas and he have helped me to develop the research skills. I would also like to thank him and the department for funding me for various events such as a summer school and conferences during my PhD.

My sincere gratitude goes to Project 911 - Vietnam International Education Department (VIED) Scholarship which funded me for three years of my PhD. Without VIED scholarship, I would not have had the opportunity to come to UK and do my PhD at University of Leeds.

Also a big thank you to all current and ex PhD students, Aryana Tavanai, Christiana Panayi, Leo Pauly, Rebecca Stone and so many others who have discussed ideas and helped me over the last four years. My special thanks go to Duane Carey and Fouzhan Hosseini who have advised and encouraged me through the hardest times. I would also like to thank the staff in the School of Computing for a lot of fruitful discussions and support, and for providing a conductive environment to research. There are many others, too many to list here, but my gratitude goes to everyone who has helped me or been a friend to me over the years. I thank you all.

Most importantly however I would like to thank my parents and family in Vietnam. Without the never ending support and encouragement of my parents, I would not be here now. I would also like to thank my husband, Nguyen Van Quyen, who has supported me at the toughest times, understanding when I was so caught up in my work and encouraging me when things went wrong. My special thanks go to my lovely son, who has brought so much joy laughter and happiness into my PhD.

Contents

List of Figures

List of Tables

List of Notations

The following is a list of important math notations used in the thesis. In general, the following rules are used for numbers and arrays:

- Bold capital letters (e.g. $\mathbf{W}$) denote matrices.
- Bold small letters (e.g. $\mathbf{w}$) denote column vectors. A row vector is denoted by its transpose, e.g, $\mathbf{w}^T$.
- Non-bold letters (e.g. x, l, C) are for scalars.

Latin

a - Negative slope in leaky ReLU layer

$\mathbf{b}$ - Network bias

$\mathbf{b}_l$ - Network bias of layer l

C_l - The depth of output tensor of layer l

C_0 - The depth of input tensor

$\mathbf{d}$ - Feature representation of a patch

$\mathbf{E}$ - Output tensor

e - Prediction/Reconstruction error

$\mathcal{F}_t$ - A video frame at time t

$\mathbf{P}$ - Foreground patch

$\mathbf{P}_n$ - n-th foreground patch

$\hat{\mathbf{P}}$ - Estimation of the foreground patch

H_l - The height of output tensor of layer l

H_0 - The height of input tensor

N - Batch size

$\mathbf{w}$ - Decision hyperplane normal vector

r - Regularity score

s - Anomaly score

thr - A threshold for anomaly score

W_l - The width of output tensor of layer l

W_0 - The width of input tensor

$\mathbf{W}_l$ - Network weights at layer l

$\mathbf{W}$ - Network weights

(x, y, c) - The row, column and channel indices of an element in the tensor

Greek

λ - Regularization term or weight decay

α_i, β_i - Lagrangian multipliers

ξ_i - Slack variables in one-class Support Vector Machine

ν - One-class Support Vector Machine parameter

ρ - Bias

γ - Radial basis kernel function parameter

υ - Intersection over Union threshold

τ - Temporal window

$\boldsymbol{\theta}$ - A video volume

Functions

g, f - Activation function

$\sigma(x)$ - Logistic sigmoid, $\frac{1}{1+\exp(-x)}$

Φ - Feature projection function

k - Kernel function

$\mathcal{L}$ - Loss function

$||\mathbf{W}||_F^2$ - Frobenius norm of $\mathbf{W}$

$||\mathbf{W}||_2^2$ - L_2 norm or Euclidean norm of $\mathbf{W}$

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ - Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T\mathbf{b}$ - Dot product between column vector $\mathbf{a}$ and $\mathbf{b}$

List of Acronyms

AMHOF - Adaptive Multi-scale Histogram of Optical Flow

AE - Autoencoder

CAE - Convolutional Autoencoder

CNN - Convolutional Neural Network

Conv-WTA - Convolutional Winner-Take-All

ConvLSTM - Convolutional Long Short Time Memory

CRF - Conditional Random Field

GMM - Gaussian Mixture Model

HOF - Histogram of Optical Flow

HMM - Hidden Markov Model

KL - Kullback-Leibler

LDA - Latent Dirichlet Allocation

LSTM - Long Short Time Memory

MHOF - Multi-scale Histogram of Optical Flow

MPPCA - Mixture of Probabilistic Principal Component Analyser

MDT - Mixture of Dynamic Textures

MRF - Markov Random Field

OCSVM - One Class Support Vector Machine

PCA - Principal Component Analysis

ReLU - Rectified Linear Unit

RNN - Recurrent Neural Network

Chapter 1

Introduction

The world around us is monitored by millions of CCTV cameras; according to a report from the British Security Industry Association in 2013, there were between 4 million and 5.9 million surveillance cameras in the UK alone. These surveillance cameras bring a significant growth of video data, increasing pressure on conventional video monitoring and analysis processes, which are usually highly labour-intensive and costly. Because of this, the need has arisen for automatic monitoring of video streams to minimise the requirement for human supervision.

Motivated by the growth in computational speed and memory capacity of computers, the industry and academics have developed key techniques for intelligent surveillance. Given a large amount of video collected by a set of surveillance cameras, an intelligent video surveillance system aims at detecting and tracking objects of interest over time, and further analysing and understanding the visual events of the scene. It has a wide range of applications, such as health care [20], traffic monitoring [21] and threat detection [22, 23].

Recent advances in camera hardware and network communication technology have led to widespread installation of video surveillance systems in private and public places, such as airports, railway stations, schools, shopping malls, hospitals and the home. Although the systems used in public places often have to deal with a high density of objects, leading to greater processing challenges, the research on crowded scene analysis has already attracted significant attention along with the increasing concern on public security and safety [24, 25].

Anomaly detection in video is a key approach to scene analysis. This approach aims at automatically identifying abnormal events within video stream, serving a wide range of applications, from monitoring surveillance cameras, or suggesting frames of interest that need to be analysed manually, to drawing the viewer's attention to salient behaviour in a video. Although automatic anomaly detection has attractive potential, it still faces lots of problems.

1.1 Challenges

In general, there are a few challenges for anomaly detection in video surveillance:

- In contrast to action recognition where events are clearly defined, the definition of anomalies in video can have some degree of ambiguity. Anomalies usually cover a wide range of activities. In addition, the definition of an anomaly changes in different applications and datasets. In some cases, different authors even define different anomalies as ground truth on a common dataset.
- The availability of labelled data, which are used to train or to validate models for anomaly detection, is usually a major problem. In practice, it may be easy to provide sufficiently many samples of normal activities, however it is difficult to provide all possible examples and types of abnormal activity that can happen in the scene. As a result, it is difficult to train a model in a supervised manner to separate an abnormal class from a normal class.
- Crowded scenes involve many independently moving objects that occlude each other in complex ways. Moreover, these videos often have low resolution. Although object tracking and trajectory analysis are considered as sufficient methods to capture high level features, they were mainly designed for scenes with low density of population. Therefore, many low level features were proposed to deal with crowded scene anomaly detection (covered in detail in chapter 2). However, most of the proposed methods for feature representation are based on hand-crafted features. It is challenging to decide which kind of feature is suitable for a specific situation.
- Low level features were built on small 2D or 3D patches extracted from video and then these features are used to learn a model of normality. As a result, anomalies are dependent on the scale at which a model of normality is defined. A normal behaviour at a fine visual scale may be perceived as highly anomalous when a large scale is considered, or vice versa. Moreover, in scenarios where scale changes significantly, features of the same object at different locations may vary due to the perspective of

the camera. An anomaly at a further distance to the camera may be ignored due to the small scale.

- An anomaly detection system is supposed to provide real-time and automatic notifications when an anomaly appears in the scene. Therefore, computational complexity and running time need to be considered.

1.2 Motivation

Although anomaly detection can be extremely challenging, this has motivated a great diversity of application and a surge of interest. Many methods have been proposed to deal with the above challenges in which anomalies are defined as events that have not been seen or have rarely been seen in the training data.

Given a set of training data, anomaly detection methods, based on low-level features, aim to learn feature representations that capture normal motion and appearance from either 2D image patches or 3D video volumes. These features of training data are then used to train a model of normality and anomalies are detected by identifying patterns which are outliers of the model. The model can be probabilistic [1, 4, 5, 6, 17, 18, 26, 27, 28, 29], a sparse coding [2, 16, 30, 31, 32] or one-class Support Vector Machine [33, 34, 35, 36] (covered in detail in Chapter 2). These methods have been shown to perform well on a variety of datasets. However, the limitation of these methods is the use of hand-crafted features, which are designed with some prior knowledge of the domain.

Recently, deep learning, especially the use of deep convolutional neural networks (CNNs) have achieved state of the art performance in a range of tasks, including object recognition [37], detection [38] and segmentation [39]. A deep CNN model learns a hierarchical set of features through multiple layers [40], each layer in the model captures information at different scales, for example low-level edges, mid-level edge junctions, high-level object parts or a complete object. Moreover, feature representations can be learned automatically from unlabelled data thus avoiding a lot of time-consuming engineering. These unsupervised feature learning algorithms are based on building an autoencoder (reviewed in detail in Chapter 2), which maps an input to a hidden feature space and then maps the features back to the input space. This is done by minimizing an error between a target and a network's output. Learning features in this way is very useful for datasets with few labels.

Deep learning has also been applied to video feature learning in a supervised setting [9, 41, 42] or an unsupervised setting [43, 44]. Spatio-temporal dynamic features in video can be exploited by the use of 3D convolution networks [9], different temporal/temporal fusion strategies [45] and Recurrent Neural Networks (RNNs) [43, 46].

Even though a deep model contains many layers with many matrix calculations, the recent availability of powerful parallel machines (i.e., GPU, CPU clusters) helps to speed up the running time.

1.3 Aims and Objectives

Given the issues and the motivation outlined above, it is the intention of this work to provide a framework to learn spatio-temporal feature representations for anomaly detection in video, with a focus on the datasets containing dense crowds.

Our main research questions hence are as follows:

- Can we train a deep learning model to automatically learn feature representations that capture the motion information in crowded video?
- Using the above feature representations, can we develop an effective method to learn normality model for anomaly detection?
- Can we build an end-to-end trainable prediction network for anomaly detection?

The work aims to solve the problem of hand-crafted feature representations by the use of an autoencoder framework in deep learning. The discriminative features can be learned automatically via multiple layers with non-linear activation functions in the autoencoder. Having a trained autoencoder, the encoder can be used to form a feature extractor in the anomaly detection system. A model of normality is trained on extracted feature representations in a training phase, which is then used to detect anomalies on features of testing data.

The work also aims to solve the problem resulting from the variation in scales as object moves in depth through the scene. The scene is divided into regions and a separate model of normality is learned at each region. Figure 1.1 describes the overview of our method that uses a convolutional autoencoder to learn motion representations. The details of this method will be presented in Chapter 3.

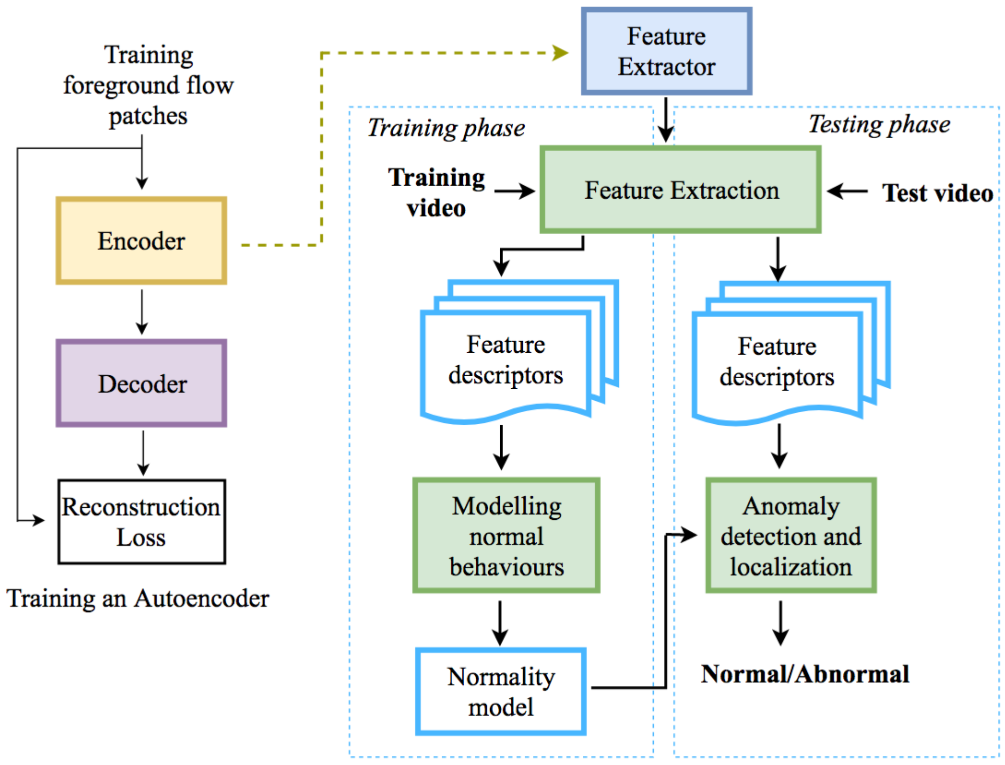


Figure 1.1: Overview of our method that uses a convolutional autoencoder to learn motion representations for anomaly detection. The details of this method will be described in Chapter 3.

When using a trained encoder to build a feature extractor, we also need to train another model on top of these feature representations for modelling normal behaviours. Normality modelling is not jointly optimized with the autoencoder. Intuitively, an encoder-decoder is trained to estimate normal samples in training set with low cost, it should estimate an abnormal samples with higher cost. Using this intuition, we aim to train encoder-decoders to learn spatio-temporal dynamics from training normal video volumes and then use an error between network's output and a target to detect a test volume as normal or abnormal. This network can be trained end-to-end without requiring any other modelling. Figure 1.2 shows the overview of our method that uses prediction error for anomaly detection. The details of this method will be presented in Chapter 4.

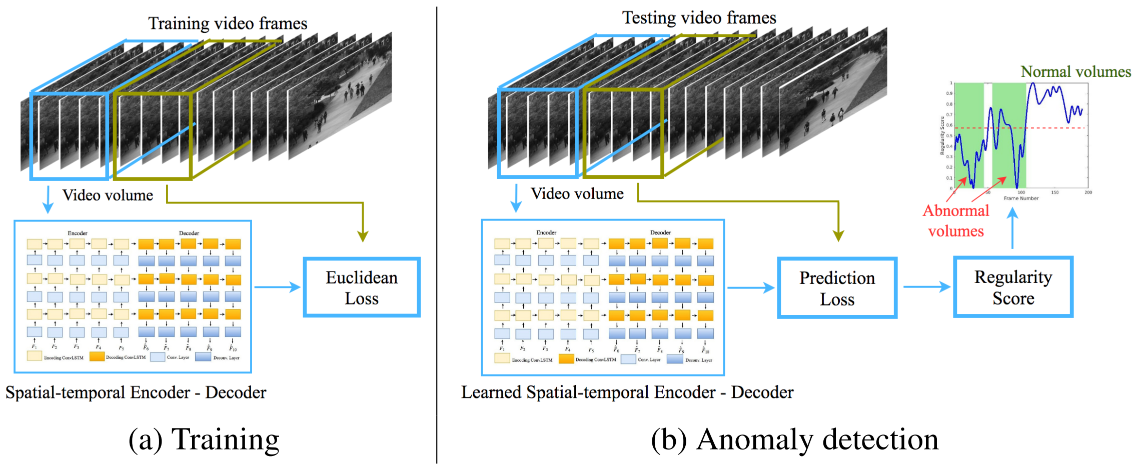


Figure 1.2: Overview of our method that uses prediction error for anomaly detection. The details of this method will be described in Chapter 4.

In our methods, we use unlabelled training data containing mostly normal samples for training autoencoders, encoder-decoders and modelling normalities.

1.4 Novelty and Significance

This thesis introduces contributions in unsupervised feature learning and normality modelling for anomaly detection in video. The following include the novel and significant contributions of this work.

- We present a novel method that uses a convolutional autoencoder to learn motion representations on foreground optical flow patches. The sparsity constraint, known as Winner-Take-All (WTA), is combined with the autoencoder to promote shift-invariant and generic flow features that are potentially discriminative for the model learning and anomaly detection using one-class Support Vector Machine (OCSVM).

- We propose an end-to-end framework that learns normal spatio-temporal dynamics from sequences of successive frames. This is done by interleaving convolutional Long Short Term Memory (LSTM) based RNNs between convolutional layers to encode temporal information on hierarchical spatial representations from low-level to high-level. To evaluate the efficiency of convLSTMs in learning temporal dynamics, we also introduce a convolutional encoder-decoder with skip connections.

- Beside the use of reconstruction error, we propose to use prediction error for anomaly detection. We train the encoder-decoders for future prediction and shows that using prediction error gives superior performance than using reconstruction error.

- We evaluate the methods on different anomaly detection datasets and demonstrate competitive performance with state of the art approaches.

1.5 Limitations and Constraints

In the proposed methods, we have a few limitations and assumptions.

- In Chapter 3, we look for anomalies via dense optical flow of successive video frames. We assume that anomalies are only found in foreground regions where there is non-zero optical flow in the image plane. The foreground patches are extracted by comparing their accumulated optical flow squared magnitudes with a threshold. Therefore, we do not attempt to detect anomalous appearance of static objects.
- We only learn (or extract) dynamic information over a range of $5 - 10$ frames. Learning longer range dynamic information can be explored in future works.
- In the work using encoder-decoders, we use an intuition that the encoder-decoder which is trained to minimise reconstruction/prediction errors on normal data, it should reconstruct/predict abnormal data with high errors. Anomalies arise as failures in reconstruction or prediction.

1.6 Outline

The rest of the thesis is organised as follows:

Chapter 2: Related Work
In this chapter we provide a literature review on some fundamental spatial-temporal feature descriptors and generative/discriminative modelling methods that have been used in anomaly detection. We also present related works within the application of deep learning methods. We primarily focus on autoencoders and their use for anomaly detection. Finally, we describe the evaluation methodology and anomaly detection datasets. Since there are many datasets, we focus only on datasets and evaluation measures that we use for performance evaluation.

Chapter 3: Convolutional Winner-Take-All Autoencoder for anomaly detection
In this chapter, we present a method using a convolutional WTA feature extractor and

OCSVM for anomaly detection. We compare architectures with the use of a sparsity constraint in training a convolutional autoencoder and without the use of it. We also present an approach to extract a smoothed motion feature representation by the combination of a trained encoder, a max pooling layer and a temporal averaging step. We show the improvement of the method that uses a convolutional WTA feature extractor over the use of a convolutional feature extractor. We also compare the performance of the framework against various state of the art approaches on two datasets.

Chapter 4: Convolutional Long Short-Term Memory for anomaly detection
To accomplish the goal of using an end-to-end trainable deep network for anomaly detection, this chapter describes a framework that interleaves convLSTM based RNNs between convolutional layers to learn spatio-temporal dynamics of normal video volumes. To evaluate the efficiency of RNNs layers, we also describe a method that uses convolutional encoder-decoders. This chapter includes details of different network architectures and approaches that use reconstruction error and prediction error for anomaly detection. We present quantitative and qualitative results and compare the method against state-of-the-art methods on three challenging datasets.

Chapter 5: Conclution and Future Work
The final chapter provides a summary of the work presented in the thesis and a final conclusion on their novelty and significance. Moreover, we also provide possible extensions and future research directions.

Chapter 2

Related Work

2.1 Introduction

Detecting unusual activities, uncommon behaviours or irregular events in video has become an important problem that has drawn a significant amount of attention in the field of automated video surveillance systems. Monitoring a large number of surveillance video streams is a cumbersome and error prone process. Automatic anomaly detection allows to reduce the amount of data needed to be processed manually by raising human attention on the specific time of anomalies appearing in a video.

Many researchers have focused on anomaly detection in video, and a comprehensive survey of this problem can be found in several review papers [47, 48, 49, 50]. Over the years, previous approaches proposed for this problem changed from rule-based methods [51, 52] to machine learning approaches. In rule-based methods, the rules are defined using prior knowledge. For example, Dee and Hogg [51] built a model of intentional, goal-directed behaviour based on an understanding of the way people navigate toward a goal. People usually move directly and purposefully to their desired destinations and these then consistently explain their behaviours. Therefore, behaviour patterns, that are not consistently explainable by the model, are abnormal. The rules can be defined on a set of events to form hypothetical explanations for the normal activities [52]. These methods work well in restricted domains and do not need (or need less) training data.

However, the rules need to be predefined, which does not scale well to new domains and may be impractical for complex behaviours involving multiple actors. Therefore, unsupervised methods are more appealing because they attempt to learn models of normality without the aid of human intervention. In this thesis, unsupervised learning is used on training data that is largely composed of normal data. As a result, we describe in this chapter unsupervised or semi-supervised methods for anomaly detection.

Most unsupervised approaches consist of two key steps, (i) feature representation and (ii) normality modelling. In this chapter we provide a literature review on these two steps. Section 2.2.1 presents the related work on spatio-temporal feature descriptors and then Section 2.2.2 reviews some generative modelling methods that have been employed for anomaly detection. Discriminative methods for modelling normality are presented in Section 2.2.3.

As mentioned in Chapter 1, deep convolutional neural networks can learn a hierarchical set of features through multiple layers and this can be done in an unsupervised manner by the use of autoencoders. Therefore, in this chapter we also highlight the successful trends that have leveraged deep learning on anomaly detection in Section 2.3. Firstly, the background on a basic autoencoder and its variants are presented in Section 2.3.1. Then, we review the methods that use deep models for feature representations, which are then combined with generative or discriminative modelling methods. These models can be pre-trained models on a large-scale image dataset for object classification (e.g. ImageNet dataset) or models trained with an autoencoder, which are described in Section 2.3.2. Finally, deep learning architectures that require no additional modelling method [12, 13, 14, 53] are described in Section 2.3.3.

Section 2.4 reviews the evaluation measures that are commonly used in current day anomaly detection. Finally, we describe briefly three challenging datasets for anomaly detection.

2.2 Non-deep learning methods for anomaly detection

Most video based anomaly detection approaches involve a feature extraction step followed by the application of traditional machine learning methods. There are mainly two approaches to extract robust and descriptive features, which capture the unique properties of normal behaviour.

The first method captures pixel-level descriptions with primitives such as gradient, colour, texture and motion. The features used to represent normality can be either global or local. They also can be either spatial or temporal or both. Among these features, spatial-temporal features have shown particular promise in motion understanding and are widely used as features descriptors [1, 2, 4, 18, 54, 55].

The second method is object-level description with primitives such as trajectory, size, shape and speed of the object. Object-based representations provide direct object-level semantic interpretation of behaviour, so approaches are performed based on detection and segmentation. Some well-known methods in this category are based on video parsing [56, 57] or object tracking [58, 59, 60]. Typically, anomaly detection is based on the use of videos from stationary cameras and object candidates can be detected with a robust background subtraction algorithm [61]. Then anomaly detection can be seen as the task of video parsing.

In the work by Antic and Ommer [56, 57], a short list of object hypotheses was computed using background subtraction and a linear SVM classifier that was trained on background and normal foreground segments. During training, normal object prototypes were learned on a set of normal objects that best explained the foreground. Then an anomaly was a hypothesis which was necessary to explain the foreground but fitted to a normal object prototype with low probability. The authors used shape, appearance and motion of normal objects to build normal prototypes.

Other works [58, 59, 60] modelled normal trajectories, which were collected by tracking individual moving objects in the video. Then an object with trajectory deviating from the normal model was detected as an anomaly. While capable of identifying abnormal behaviours, video parsing and tracking are both difficult and computationally expensive for crowded and complicated scenes.

Therefore, when taking the above into consideration, this review of non-deep learning methods only focuses on hand-craft feature descriptors that are extracted from appearance in still image data or motion patterns in Section 2.2.1. Generative and discriminative models that have been adopted for anomaly detection are reviewed in Section 2.2.2 and Section 2.2.3.

2.2.1 Feature descriptors

Appropriate feature descriptors benefit the subsequent normal model building in anomaly detection. In recent years, a number of spatio-temporal feature descriptors have been proposed in video processing, containing texture and/or motion information.

Boiman and Irani [1] densely extracted spatio-temporal patches from video at various spatial and temporal scales. For each small patch, a descriptor vector was computed and stored, along with the absolute spatial-temporal coordinates of the patch. The descriptor of each video patch was constructed from the absolute values of the temporal derivatives in all pixels of the patch. To account for both local and global compositional information in video regions, the authors broke down a large region into an ensemble of hundreds of small spatio-temporal patches at multiple scales with their relative geometric positions. An example of the ensemble of patches is illustrated in Figure 2.1. The approach is invariant to small changes in local parts of a configuration. This is done by allowing for small local misalignments in the relative geometric arrangement when searching for similar ensembles.

The primary drawback of this work is dense sampling. Dense sampling at different scales yields a large number of patches which faces scalability issues. Moreover, densely sampled patches are also redundant once training data is very large. Therefore, similar patches were grouped by constructing a codebook using the 'bag of video' approach [54].

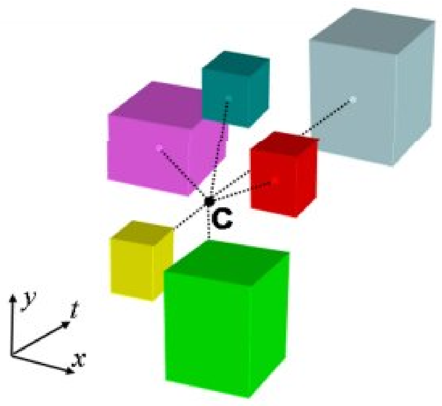


Figure 2.1: An ensemble of patches in video, **c** is an origin of the ensemble. Figure from [1].

Many other works have made use of local image features and these include the use of histograms of optical flow [2, 3, 33], histograms of pixel change [62], measures based on optical flow [4, 18] and 3D Gradient [5, 16, 63].

Optical flow [64, 65] is the pattern of apparent motion of objects, surfaces and edges between two consecutive frames caused by the movement of objects or the camera. It is a 2D vector field where each vector is a displacement vector showing the movement of scene points from first frame to second. Descriptors can be built on optical flow to capture the long-range spatial and temporal properties.

For example, in order to construct a feature descriptor representing a 2D image region, Kim and Grauman [18] divide the region into $u \times v$ sub-regions (so-called 'units'). Each unit is represented by a 9 dimension vector computed from optical flow. The first eight bins of the vector correspond to a histogram of eight optical flow orientation ranges and the last bin is the sum of optical flow magnitudes of all pixels in the unit. Finally, a $9 \times u \times v$

dimensional descriptor for the image region is constructed by concatenating the descriptors of all units.

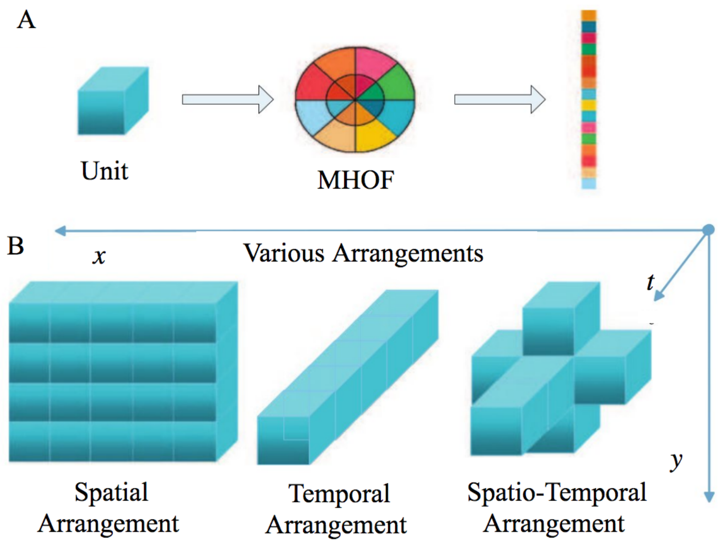


Figure 2.2: (A) Multi-scale Histogram of Optical Flow [2] is extracted from a basic unit. (B) The selection of spatial-temporal basis used for anomaly detection. Figure reproduced from [2].

The optical flow magnitude of each pixel in the region can be voted into one of d bins using its optical flow direction to form a d-bin histogram (so-called Histogram of Optical Flow - HOF). This simple histogram of optical flow descriptor has been extended to describe complex motions in anomaly detection [2, 3, 33].

Cong et al. [2] used eight bins for eight ranges of direction, combining with two scales of magnitude to create a 16-bin histogram (so-called Multi-scale Histogram of Optical Flow (MHOF) descriptor). Applying a threshold on flow magnitude, the first eight bins denoted direction ranges with motion magnitude less than the threshold, and the second eight bins denoted direction ranges with magnitude equal or greater than the threshold. MHOF descriptor was built on several arrangements of units as illustrated in Figure 2.2 to handle different abnormal events: the group behaviour of global scene (global abnormal event) or the behaviour of an individual in the scene (local abnormal event). 16-bin MHOFs of all units in the arrangement are calculated and are then concatenated to form a final descriptor. Different arrangements capture different information such as spatial, temporal or spatio-temporal information. While stacking multiple units at the same location over time helps to incorporate temporal information, concatenating neighbouring units preserves spatial contextual information.

The 16-bin MHOF was extended to a histogram of $m \times n$ bins (so-called Adaptive Multi-scale Histogram of Optical Flow (AMHOF)) by Lin et al. [3] where m is the number of directions and n is the discrete level of the motion intensities. Instead of using a fixed partition of motion magnitudes defining the n bins, the bin edges was determined using k-means clustering on a random subset of magnitude of flow vectors from the training data. To use this descriptor for anomaly detection, the authors [3] stacked successive units at the same location, then a histogram of $m \times n$ bins was extracted. This descriptor considers the temporal relation between units in a similar way to Cong et al. [2]. However, the histogram is computed from optical flow of all pixels in the arrangement, instead of concatenating histograms of separate units.

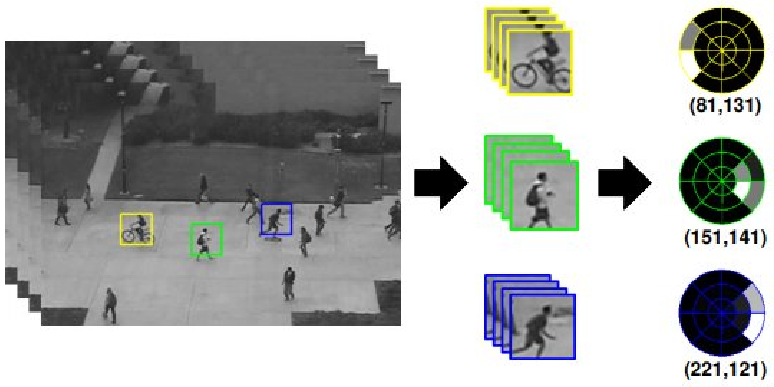


Figure 2.3: AMHOF descriptors of three regions of interest. Figure from [3].

Another variant of histogram of optical flow was proposed to capture local motion information of foreground objects in the work by Wang et al. [33]. A spatio-temporal cuboid was partitioned spatially into smaller spatio-temporal regions. Then HOFs, which were separately extracted from these small regions, were concatenated into a descriptor for the cuboid. According to Wang et al. [33], this descriptor can capture the difference in local motions. For example, consider a walker and a skater moving in the same direction with similar speeds, the HOF descriptors of the upper parts of their bodies are the same, however, the HOFs for the person's legs are different from the skater.

The social force model describes the behaviour of a crowd as a result of interaction of individuals. This method has been applied for pedestrian motion dynamics by considering internal motivations of individuals to perform certain movements and environmental constraints [66]. The actual force consists of the internal desire force and the interaction force. To estimate interaction forces for anomaly detection without object tracking, a crowd is treated as a collection of interacting particles [4, 27] (Figure 2.4). A grid of particles is overlaid the image in which particles move with the underlying flow field. The optical flow is used for estimating interaction forces. In particular, the average of the optical flows over a fixed window of space and time around a particle is used as the actual velocity of the particle. Moreover, each particle has a desired velocity, which is computed from the corresponding optical flow that particle overlays. The interaction force of the particle with the surrounding particles or the environment is estimated using the difference between its desired velocity and its actual velocity. Finally, the magnitude of the interaction force vectors is mapped to the image plane to construct a feature matrix of force flow for the image. Spatio-temporal cuboids are extracted from these feature matrices over successive video frames and then are used to learn a generative model (using the Latent Dirichlet Allocation method, for example) for anomaly detection [4, 27].

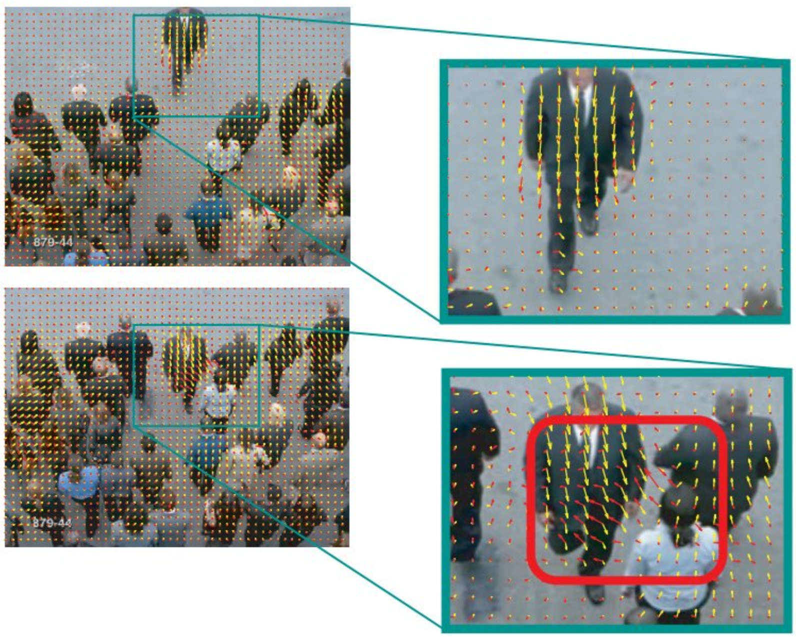


Figure 2.4: The interaction force (red) of two sampled frames is calculated based on optical flow (yellow). Figure from [4].

Kratz and Nishino [5] have proposed a 3D Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to model the distribution of spatio-temporal gradients in a local video volume with N pixel values:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i}^{N} \nabla \mathbf{I}_i, \quad \boldsymbol{\Sigma} = \frac{1}{N}\sum_{i}^{N} (\nabla \mathbf{I}_i - \boldsymbol{\mu})(\nabla \mathbf{I}_i - \boldsymbol{\mu})^T \tag{2.1}$$

where $\nabla \mathbf{I}_i$ is the spatio-temporal gradient of pixel i:

$$\nabla \mathbf{I}_i = [\mathbf{I}_{i,x}\mathbf{I}_{i,y}\mathbf{I}_{i,t}]^T = \left[\frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial t}\right]^T \tag{2.2}$$

For each spatial location l and temporal location t, the local spatio-temporal motion representation is defined by $\boldsymbol{\mu}_t^l$ and $\boldsymbol{\Sigma}_t^l$. This multivariate Gaussian modelling presents the range of motions observed in the cuboid.

Instead of using a Gaussian distribution, the spatio-temporal gradient $\nabla \mathbf{I}_i$ (so-called 3D-Gradient) has been used as a descriptor for anomaly detection in other works [16, 63]. By using this, the descriptor is simpler, but the dimension of 3D-Gradient features depends on the cuboid size. To deal with this problem, Principal Component Analysis has been employed to reduce the dimension of the representation [16, 63].

Beside the motion-related descriptors, appearance has also been combined with motion to detect anomalies [6, 17]. Dynamic texture [67] is an efficient method to consider both appearance and dynamic in the video sequence. Specifically, the dynamic texture is a generative model. It consists of a random process containing an observed variable $\mathbf{y}_t$, which encodes the appearance component (video frame at time t), and a hidden state variable $\mathbf{x}_t$, which encodes the dynamics (the change of the video over time). The relations between the states and observed variables are represented through the linear dynamic system as follows:

$$\begin{cases} \mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{v}_t & \mathbf{v}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_1) \\ \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{u}_t & \mathbf{u}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_2) \end{cases} \tag{2.3}$$

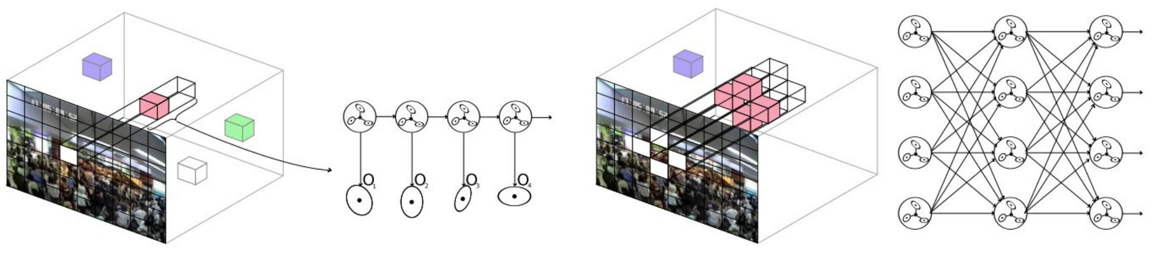
where $\mathbf{x}_t \in \mathbb{R}^n$, $\mathbf{y}_t \in \mathbb{R}^m$ $(n \ll m)$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a state transition matrix, $\mathbf{C} \in \mathbb{R}^{m \times n}$ is an observation matrix. $\mathbf{v}_t \in \mathbb{R}^n$ and $\mathbf{u}_t \in \mathbb{R}^m$ are the independent and identically distributed sequences drawn from Gaussian distributions with zero mean and covariance $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, respectively. Many methods can be used to learn the parameters of the dynamic texture from a video sequence, including maximum likelihood [68], subspace identification algorithm [69] or a closed-form sub-optimal solution for computational efficiency [67]. The dynamic texture mixture model has been considered as being more suitable for local unusual event detection in crowded scenes than by the use of optical flow only [6, 17].

In the anomaly detection methods without tracking, where motion and texture are employed, various motion-based hand-craft feature descriptors have been proposed and have shown state-of-the-art results. These above descriptors were designed to combine with different methods for modelling normality and inferring an anomaly. In the next section, the generative models applied to the above feature descriptors are described.

2.2.2 Generative models of descriptors

A standard approach to detecting anomalies in video is to estimate a generative probability distribution from the set of descriptors extracted from the training videos. Descriptors extracted from a test video are then labelled as abnormal if they are outliers of the distribution. The distribution can be learned using either a parametric [1, 4, 5, 6, 17, 18, 26, 27] or non-parametric approach [28, 29].

Often the generative model is over spatial and temporal arrangements of descriptors, for example using an Hidden Markov model (HMM) to model sequences of descriptors at a fixed position in the image plane [5, 26]. Kratz and Nishino [5] used a 3D Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as a feature descriptor for a spatio-temporal cuboid. Kullback-Leibler (KL) divergence is then used as a distance metric in online clustering to build normal prototypes. Given a local spatio-temporal cuboid, the probability of it belonging to a specific prototype is estimated using KL distance. A pattern with low probability is identified as an unusual pattern. The set of prototypes illustrates normal activities in the scene, however it does not capture the relationship between motion patterns. Therefore, HMMs are combined to model spatial-temporal relationship. A single HMM is modelled at each spatial location to account for temporal dynamics. Moreover, to capture the relationship between spatially neighbouring cuboids which occur across temporal frames, a coupled HMM is constructed among surrounding tubes as shown in Figure 2.5. In both case, the prototypes obtained from clustering are used as hidden states, hence the emission probability of the HMM models corresponds to the probability of an observation belonging to a prototype.



(a) (b)

Figure 2.5: (a) Distribution based HMM and (b) coupled HMM for capturing temporal and spatial relationships. Figure from [5].

Generative models can be used to present a set of descriptors and their conditional dependencies, for example the use of a Bayesian network to simultaneously impose constraints on the relative geometric arrangement of patches in an ensemble as well as their descriptors [1]. For a new observation, a large region surrounding every pixel is represented by an ensemble of hundreds of small patches at multiple scales with their relative geometric positions to the center of the ensemble. Using a Bayesian network, Boiman and Irani [1] solved the anomaly detection problem by computing the joint likelihood that the observed ensemble was similar to some ensemble in the database. A high likelihood means that the observed ensemble can be composed by a large number of patches in the database thus it is normal, otherwise it is abnormal. The authors considered both the similarity in descriptors of patches as well as the similarity in their relative positions. The inference method searched for a hidden database ensemble, which maximized the joint likelihood using a belief propagation algorithm [70]. This method showed good performance in detecting anomalous behaviour. The main drawbacks of this method are high computation cost for the searching process and high memory for storing patch descriptors.

A generative probabilistic model of a corpus, such as Latent Dirichlet Allocation (LDA) [71], has been employed to estimate the distribution of topics for normal crowd behaviour [4, 27]. The trained model is used to estimate the likelihood of being normal for blocks of T frames. To use LDA, a video sequence of T frames (a clip) is likened to a document with spatio-temporal cuboids as visual words. Though LDA has been applied on spatial-temporal cuboids extracted from force flow in some works [4, 27], other spatio-temporal descriptors can be used with LDA. LDA is shown to be very effective in detecting a global abnormal event (i.e., abnormal event in the whole clip), it is still challenging to localize an abnormal event. Employing LDA modelling with force flows [4], anomalies are expected to occur in the regions with higher social interaction. Therefore, areas with high force flow in an abnormal clip are considered to localize anomalies [4].

Other probabilistic methods for modelling normal behaviours are mixture models [6, 17, 18]. These methods can be used on spatio-temporal descriptors to build local normal models, then other methods (such as, Markov Random Field) can be combined to account for the spatio-temporal relationship between local patterns. For example, Kim and Grauman [18] used a simple descriptor which was a 9-dimension vector extracted from flow field on a local region and then applied a mixture of probabilistic principal component analyser (MPPCA) algorithm [72] to learn a generative model for these local patterns.

However a local model could not capture the abnormal interactions between multiple local patterns, therefore a space-time Markov Random Field (MRF) was adopted to account for spatio-temporal interactions. From the learned MPPCA model, the frequency histogram at each node and the co-occurrence histogram at each link between two nodes were computed to establish the space-time MRF. The frequency histogram considers the frequency of each component observed at each location, while the co-occurrence histogram represents the frequency of two component appearing at neighbouring nodes. Given a new observation, inference on the MRF graph, the maximum posterior probability specifies which local regions are normal or abnormal. Because the method is built on a flow based descriptor, it only emphasizes motion dynamics.

Based on the efficiency of a dynamic texture model for both appearance and dynamics of video sequences, a mixture of dynamic textures (MDT) [73] has adopted to model temporal and spatial normalities [6, 17]. The MDT models a sequence of τ video frames as a sample from one of K dynamic textures.

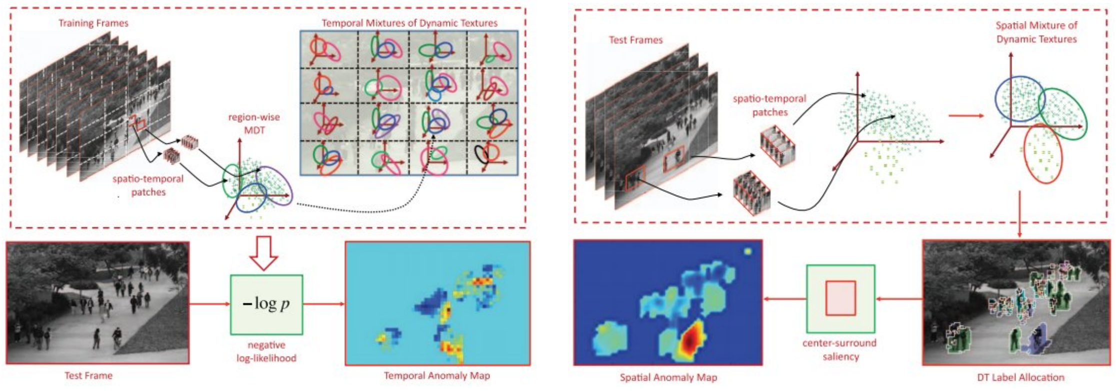
$$p(\mathbf{y}_{1:\tau}) = \sum_{i=1}^{K} \pi_i p(\mathbf{y}_{1:\tau}|z = i) \tag{2.4}$$

where $z \sim$ multinomial$(\pi_1, ..., \pi_K)$ and the mixture components $p(\mathbf{y}_{1:\tau}|z = i)$ are defined by the linear dynamic systems as in Equation 2.3. This model generates the mixture of probabilistic PCA models, as the matrix $\mathbf{C}$ in Equation 2.3 is the PCA parameter applied on patches drawn from a mixture component z.

Mahadevan et al. [17] used spatial-temporal cuboids on appearance as a sequence of observations, thus the method employed MPPCA on appearance patches, not optical flow. Moreover, the dynamic motions were captured by the hidden state $\mathbf{x}_{1:\tau}$, which can be seen as trajectory in PCA space [6, 17].

Mahadevan et al. [17] employed MDT for modelling both temporal and spatial normalities as illustrated in Figure 2.6. The model of temporal normality enables the detection of behaviours that are different from the behaviours in the past while the model of spatial normality enables the detection of behaviours that are different from the surroundings. In order to model temporal normality, a scene is divide into regions. For each region, a MDT is learned during training on spatio-temporal patches extracted from that region over time. After training, patches of low probability under their corresponding MDT are considered as abnormalities.

In addition to a temporal model, spatial anomaly detection as illustrated in Figure 2.6 (b) is based on saliency detection that defines salient locations as where the features contrast with their surrounding features. Therefore, the spatial model requires no training data, a MDT is learned on a batch of frames (or a volume) around a current test frame. For a testing volume, a dense collection of spatial-temporal patches are extracted and are used to learn a single MDT with K_{global} components. Each patch in the volume is then assigned to the mixture component with maximum posterior probability. At each location l in the scene, a centre window containing the location and a surrounding window containing background are defined. Sharing components of the learned MDT for both centre and surrounding classes, the saliency of the location l is calculated by the discriminant power to classify windows into center and surround class. This is done using KL divergence between MDT components, the ratio of pixels assigned to each component in respective windows and the ratio of volumes of these two windows.



(a) (b)

Figure 2.6: (a) Temporal and (b) Spatial anomaly detection with MDT models. Figure from [6].

Li et al. [6] improved the method of Mahadevan et al. [17] by learning MDTs at different scales. For modelling temporal normality, the authors used different sizes when dividing a scene into sub-regions. Each MDT is trained for each sub-region at the fine scales, while the whole visual field is presented with a global MDT at the coarsest scale. For modelling spatial normality, the authors used different sizes of a surrounding window with a fixed size of centre window. The performance is significantly improved but with higher computational cost.

Probabilistic approaches are mathematically well-grounded and can effectively identify abnormal data if an accurate estimate of the probabilistic density function is obtained. However, in many scenarios where prior knowledge of the data distributions is unavailable, parametric approaches may be problematic if the data do not follow the assumed distribution. In the next section, we survey discriminative models that have been applied for anomaly detection.

2.2.3 Discriminative models of descriptors

We categorize sparse coding and the support vector machine into discriminative methods because they do not need to model a probability distribution. Specifically, sparse coding can be seen as a reconstruction-based method for anomaly detection. It can model the underlying data, and when test data are presented to the system, the reconstruction error, defined to be the distance between the test representation and the output of the system, can be related to an anomaly score. Working in a different way, a support vector machine requires a boundary around the normal data to be created based on the training dataset. The class of test data is then determined by its location with respect to the boundary. In this section, we describe the idea of each method and then review related works using these methods.

Sparse coding

Sparse coding [74] aims to represent input vectors approximately as a sparse weighted linear combination of "basis vectors". The basis set can be over-complete with the number of basis vectors greater than the input dimension. Thus it can efficiently capture structures and patterns inherent in the input data. The sparse characteristic of the representation is motivated by the observation that most sensory data, such as natural images, can be described as the superposition of a small number of atomic elements such as edges.

In general, given an input matrix with m samples $\mathbf{X} \in \mathbb{R}^{k\times m}$ (each column is an input vector), a dictionary of n basis vector $\mathbf{B} \in \mathbb{R}^{k\times n}$ and the coefficient matrix $\mathbf{S} \in \mathbb{R}^{n\times m}$ (each column is a coefficient vector) are estimated by solving the following optimization problem:

$$\begin{aligned} \text{minimize}_{\mathbf{B},\mathbf{S}} \quad & \frac{1}{2}||\mathbf{X} - \mathbf{BS}||_F^2 + \beta \sum_{i,j} \phi(\mathbf{S}_{i,j}) \\ \text{subject to} \quad & \sum_i \mathbf{B}_{i,j}^2 \le c, \forall j = 1, ..., n. \end{aligned} \tag{2.5}$$

where c, β are constants. The first term is a reconstruction error which tries to force the algorithm to provide a good representation of an input while the second term $\phi(.)$ is a sparsity penalty which forces the representation to be sparse. The constraint $\sum_i \mathbf{B}_{i,j}^2 \le c$ is added to prevent the case that $\mathbf{B}$ is scaled up and $\mathbf{S}$ is scaled down to get small sparsity penalty.

The problem is convex with respect to the coefficient $\mathbf{S}$ when the dictionary $\mathbf{B}$ is fixed and also convex with respect to $\mathbf{B}$ when $\mathbf{S}$ is fixed. The solution method is to alternate between the two variables, minimizing one while holding the other fixed.

Sparsity based anomaly detection models have been proposed [2, 16, 30, 31, 32] which achieve state of the art performances in many datasets. In these works, sparsity reconstruction cost is used to reflect an anomaly level. This follows the intuition that a normal event is likely to generate sparse reconstruction coefficients with a small reconstruction cost, while the abnormal event generates a dense representation with a large reconstruction cost. Sparse coding has been employed on different descriptors such as MHOF [2, 30, 32], 3D-Gradient [16] or HOG+HOF [31] to detect anomalies. These methods focus on solving the problem of learning a dictionary and/or deriving a robust reconstruction cost for anomaly detection.

Instead of alternatively optimizing a dictionary and coefficients, the dictionary is chosen from training data in the work by Cong et al. [30]. Given an initial candidate feature pool extracted from training data, the optimal subset of it is selected as a dictionary, from which the rest of the candidates can be well reconstructed [2, 30]. This dictionary selection method was designed with a sparsity consistency constraint which helped to avoid the case that all candidates are selected. Beside the use of reconstruction cost, frequency of each basis in the dictionary has also been considered. If the basis vector appeared frequently to present training data, then the cost to use it in the reconstruction should be low [2, 30].

Work by Zhao et al. [31] showed that adding a smoothness regularization to the object function (Equation 2.5) is an effective way to consider relationships between neighbouring cuboids in an event. This regularization helps to assign similar coefficient vectors to neighbouring cuboids containing similar motions. Therefore, the coefficient vectors change smoothly over space/time across actions in the event. After training, these dictionaries can be updated using newly normal observations [2, 30, 31]. Another change to the optimization was the use of earth mover's distance as a distance metric for reconstruction cost in the objective function instead of Euclidean distance in the work of Zhu et al. [32].

In order to speed up the testing time, instead of finding a dictionary of n basis vectors, K basis combinations with each containing s dictionary basis vectors ($s \ll n$) were estimated in the work by Lu et al. [16]. For each testing feature, the most suitable combination is found by checking the least square error for each combination. This method reaches a high detection rate at about 150 frames per second.

Sparse coding methods have achieved state of the art performances in many datasets. However, a significant limitation of sparse coding is that even after a set of basic vectors has been learned, optimization must be performed to obtain the encoding coefficients for a new data sample. Therefore, the sparse coding is computationally expensive at test time especially compared to typical feed-forward architectures such as an autoencoder (review in detail in Section 2.3.1).

Support Vector Machines

Support Vector Machines (SVMs) [75, 76] are a popular technique for forming an optimal decision boundary (also known as a hyperplane) that separates data into different classes.

The original SVM is ideally for binary pattern classification of data that are linearly separable (as in Figure 2.7). The optimal hyperplane maximizes a margin between two classes. By applying a kernel trick, SVM can handle nonlinear classification problems [77]. The idea is to map the training data into a higher-dimensional feature space, and then construct a separating hyperplane with maximum margin there. This yields a non-linear decision boundary in the input space.

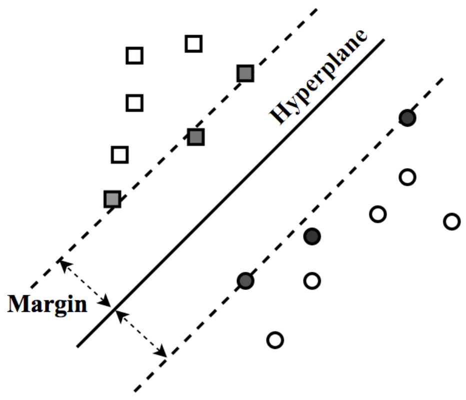


Figure 2.7: SVM for binary classification, where support vectors are the training points that lie near the hyperplane defining the margin.

Tax and Duin [78], Scholkopf et al. [76] have developed various algorithms based on SVMs to tackle one class classification problems where only one category of (the positive) samples is available. The idea is to define the boundary in the feature space corresponding to a kernel, separating the training data from the origin in the feature space with maximum margin. The origin plays a crucial role in the methods of both Scholkopf et al. [76] and Tax and Duin [78], where it acts as a prior for where the abnormal instances are assumed to lie. To explain the reason, the authors assume that a Gaussian kernel is used. With a kernel function $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}|^2}$, all dot products $k(\mathbf{x}, \mathbf{y}) > 0$ between data points in the feature space. This implies that all mapped points lie inside the same orthant (generalization of quadrant to n dimensions). Moreover, all data lie on the surface of a hypersphere in feature space since $k(\mathbf{x}, \mathbf{x}) = 1$. The objective is therefore to construct a hyperplane which is maximally distant from the origin with all data points lying on the opposite side of the origin. In practical implementations, one class SVM (OCSVM) of Scholkopf et al. and method of Tax and Duin perform comparably and both perform best with the Gaussian kernel [79].

OCSVMs have been adopted for anomaly detection with batch mode learning [33, 34] or online learning [35, 36]. The key advantage of online learning is to train an OCSVM for a large-scale dataset or streaming data in which video frames arrive sequentially rather than all at once. Different online learning methods have been proposed specially for anomaly detection [35, 36]. Lin et al. [35] used incremental and decremental OCSVM within a sliding buffer to solve the problem of increasing processing time as the dataset grows, resulting in a suitable method for a real time system. Wang et al. [36] extended an online least square OCSVM [80], where the solution was found by solving a linear system instead of solving a quadratic programming problem, with a sparse solution to adapt to online abnormal event detection.

A drawback of one-class SVM methods is the computational complexity associated with the kernel functions. Moreover, the choice of the appropriate kernel function may be problematic. Additionally, it is not easy to select values for the parameters which control the size of the boundary region.

In our work, we use one-class SVM in batch mode to train a model for normality and then use it for detecting anomalies. Rather than using hand-crafted features, an autoencoder is employed to learn a feature representation. In the next section, we review related works that use deep learning methods.

2.3 Deep learning in anomaly detection

Since the success of convolutional neural networks (CNNs) in the 2012 ImageNet competition [37], rapid progress has been made on feature learning using deep networks. Deep methods have been applied successfully to a large variety of computer vision tasks, for example to object recognition [37], object detection [38], segmentation [39], action recognition [42] and super-resolution [81]. Various pretrained convolutional network models (such as, AlexNet [37], VGGNet [82], GoogLeNet [83]) are available for extracting image features. These models are trained on a very large dataset, such as ImageNet with roughly 1.2 million training images from 1000 categories.

Donahue et al. [84] show that features extracted from the activations of a deep convolutional network trained on a large dataset for object recognition tasks can be re-purposed to novel generic tasks such as object recognition in a new domain (domain adaptation) which may differ significantly from the originally trained tasks. The key benefit is that there may be insufficient labelled data to train or to adapt a deep architecture to the new tasks.

Moreover, feature representations can be learned from unlabelled data with the use of a deep autoencoder. In this section, we present the autoencoder and the use of this method in the related works. While there are many variants of the basic autoencoder, we focus on autoencoders that have been used for anomaly detection.

2.3.1 Autoencoder and its variants

Auto-encoder

An autoencoder (AE) is the most common method that is used when considering the use of deep learning for unsupervised classification. Figure 2.8 shows a taxonomy of AE [7], each AE is designed to learn feature representations with desired properties, such as features with lower dimensionality, sparsity, noise tolerance or feature representations drawn from a distribution.

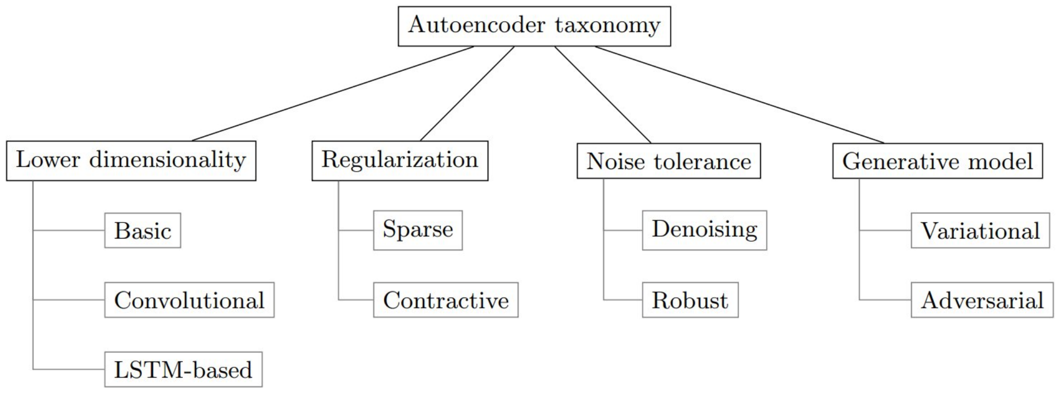


Figure 2.8: Taxonomy: most popular autoencoders classified according to the charasteristics they induce in their encodings [7].

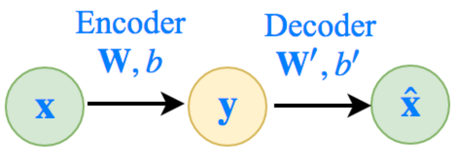


Figure 2.9: A general autoencoder structure.

In its simplest form, the autoencoder is a one hidden layer artificial neural network. The middle layer represents an encoding of the input data that presents some of the above desired properties. The basic structure of an AE, which is shown in Fig. 2.9, is trained to encode the input $\mathbf{x}$ into representation $\mathbf{y}$ via an encoder. A decoder then maps the latent representation $\mathbf{y}$ to the reconstruction $\hat{\mathbf{x}}$.

$$\begin{aligned} \mathbf{y} &= f(\mathbf{W}\mathbf{x} + b) \\ \hat{\mathbf{x}} &= f(\mathbf{W}'\mathbf{y} + b') \end{aligned} \tag{2.6}$$

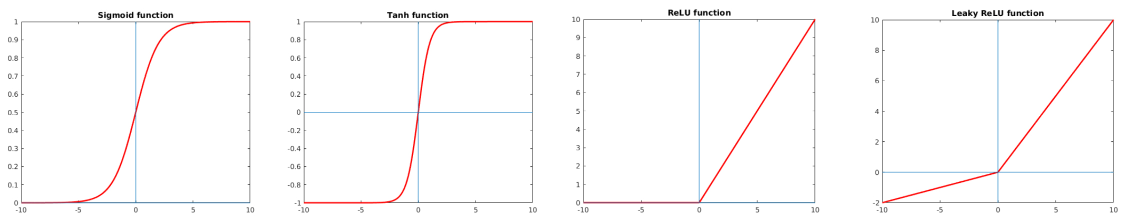
where f is a non-linear function such as the sigmoid or ReLU function (Figure 2.10).

The parameters $\mathbf{W}, \mathbf{W}', b, b'$ are optimized such that the reconstruction loss $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$ is minimized. For real-valued $\mathbf{x} \in \mathbb{R}^d$, mean squared error $\mathcal{L}(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2$ is typically used.

Activation functions of common use in autoencoders

The sigmoid function is the most common activation function in autoencoders. The output is in the range $[0, 1]$, which is not symmetric around the origin and the values received are all positive. A problem that the sigmoid function suffers is that the gradients become very small when the input values fall beyond the range $[+3, -3]$.

The hyperbolic tangent is a scaled version of the sigmoid function. It works similarly to the sigmoid function but is symmetric around the origin in the range $[-1, 1]$. Similar to the sigmoid function, hyperbolic tangent still suffers the vanishing gradient problem. According to Lecun et al. [85], the tanh should be preferred, since it often converges faster than the standard sigmoid function.



$$f(x) = \frac{1}{1+e^x} \qquad f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad f(x) = \max(0, x) \qquad f(x) = \begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases}$$

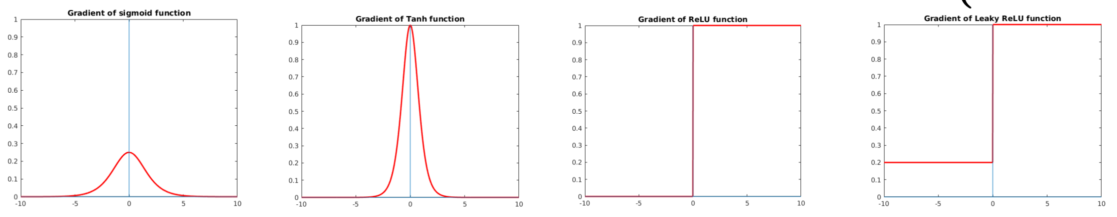
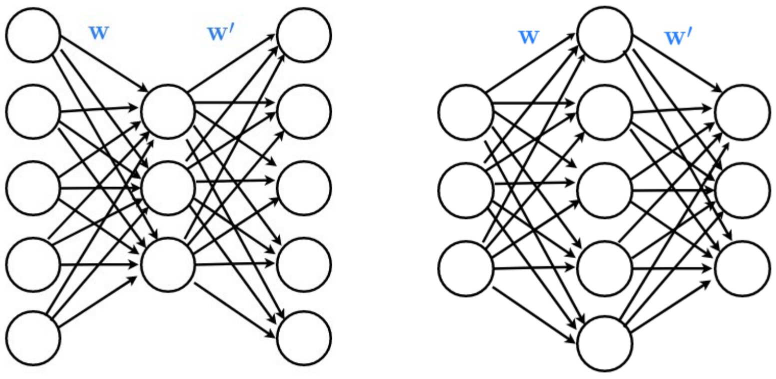


Figure 2.10: Common activation functions.

The ReLU is the rectified linear unit [37] which is one of the keys to the recent success of deep networks. According to Krizhevsky et al. [37], ReLU expedites convergence of the training procedure. It helps to solve the vanishing problem. However, with the negative input, the gradient is zero, the weights are not updated during back propagation. This can lead to 'dead' neurons, which never get activated. Variants of ReLU function have been proposed, such as leaky ReLU [86] and parametric ReLU [87]. These functions use a coefficient to control the slope of the negative part, which helps to overcome the zero gradient problem. Sigmoid, tanh, ReLU and leaky ReLU (negative slope $a = 0.2$) activation functions are shown in Figure 2.10.

Learning good representations with fully-connected autoencoders

Training an autoencoder by minimising reconstruction error is not sufficient to yield a useful representation [88]. For example, an autoencoder where $\mathbf{y}$ has the same dimensionality as input $\mathbf{x}$ can achieve a perfect reconstruction simply by learning an identity mapping. Without any other constraints, this criterion alone is unlikely to lead to the discovery of a more useful representation than the input .



(a) Under-complete autoencoder (b) Over-complete autoencoder

Figure 2.11: Under-complete and over-complete auto-encoders.

Thus further constraints need to be applied to retain useful information from noise. The traditional approach used by autoencoders is referred to as bottleneck and is shown in Figure 2.11 (a). The resulting lower dimensional $\mathbf{y}$ can thus be viewed as a lossy compressed representation of $\mathbf{x}$. An alternative architecture is to use an over-complete representation, as shown in Figure 2.11 (b), with a sparsity constraint [89]. A sparse over-complete representation can be viewed as an alternative “compressed” representation with a large number of zeros.

Denoising has also been introduced as a training criterion for learning useful features with an autoencoder [88, 90]. A denoising autoencoder, is trained to reconstruct a clean input $\mathbf{x}$ from a partially corrupted version $\tilde{\mathbf{x}}$. The corrupted inputs are obtained by (1) adding Gaussian noise or (2) randomly choosing a fixed number of components of $\mathbf{x}$ and forcing their values to 0. In this way, the autoencoder cannot learn the identity of $\mathbf{x}$, thus a denoising autoencoder does not need any constraint or regularization to learn a meaningful latent representation.

To learn a set of discriminative feature representations with a deep AE, shallow AEs are stacked successively to form a multiple layer autoencoder or a stacked autoencoder. An effective way to obtain good parameter values for a stacked autoencoder is to use greedy layer-wise training [91].

Convolutional Auto-encoder

Fully connected autoencoders ignore the spatial relations in images. Moreover, it introduces a large number of parameters when working with large images. A convolutional autoencoder (CAE) [8, 92] employs the advantage of a convolutional layer, dealing with uncompressed images with small shared weights.

The convolution naturally operates on the input of any size and produces an output of corresponding spatial size. An output is achieved by convolving the filter and the input tensor. The output size depends on the input size, kernel size, the amount by which the filter shifts (so-called stride) and the depth of zero padding around the input borders. Locations in higher layers correspond to the locations in the input image they are path-connected to, which are called their receptive fields.

The CAE architecture is similar to a basic fully connected autoencoder except that matrix products are replaced with convolutions, a 1D input is replaced with a 2D/3D tensor. For an input $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, the k-$th$ feature map $\mathbf{Y}^k$ is given as in Equation 2.7. The reconstruction $\hat{\mathbf{X}}$ is obtained by mapping the latent feature maps back to the input space as follows:

$$\begin{aligned} \mathbf{Y}_k &= g(\sum_{c \in C} \mathbf{X}_c * \mathbf{W}_{c,k} + b_k) \\ \hat{\mathbf{X}}_c &= g(\sum_{k \in K} \mathbf{Y}_k * \mathbf{W}'_{k,c} + b'_c) \end{aligned} \tag{2.7}$$

where encoding convolutional filters $\mathbf{W} \in \mathbb{R}^{H_1 \times W_1 \times C \times K}$ and decoding convolutional filters $\mathbf{W}' \in \mathbb{R}^{H_2 \times W_2 \times K \times C}$, '$*$' denotes to 2D convolution. $g(.)$ is an activation function. K is the number of latent feature maps in the hidden layer. There is a single bias b_k for each feature map and one bias b'_c per input channel.

An example of the use of a CAE on images from the "two bar" dataset [8] is illustrated in Figure 2.12. An encoder consists of a convolutional layer and a max-pooling layer, a decoder mirrors the encoder with an upsampling (also known as unpooling) layer and a convolution. Max-pooling [93] can be used on the top of the convolutional layer in the encoder to down-sample the latent representation.

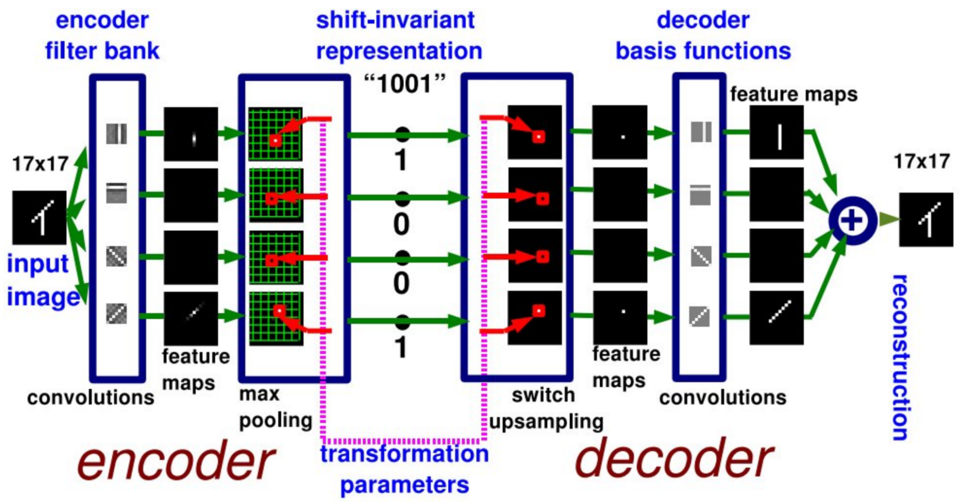


Figure 2.12: Architecture of the shift-invariant unsupervised feature extractor applied to the two bars dataset. Figure from [8].

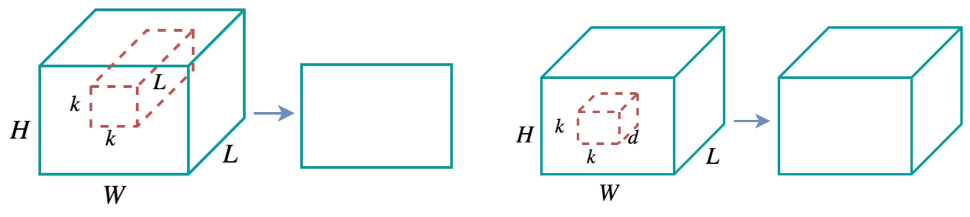
The max-pooling operation produces the maximum output within a rectangular window. Pooling helps to make the representation invariant to small translations of the input. If the input is translated by a small amount, the value of most of the pooled outputs do not change. However, spatial information within the receptive field is lost. Therefore, both value and location of the maximum are recorded, in which the feature representation contains values and transformation parameters contain the locations (Figure 2.12). The transformation parameters are then used in a corresponding unpooling layer to do up-sampling. The unpooling operation takes elements in its input and places them in its output tensor at the location specified by the transformation parameters.

The decoder transforms data in the opposite way to the encoder, thus a transpose convolution (also called a deconvolution or a fractionally strided convolution) can be considered to replace a convolution in the decoder. A deconvolution works by swapping forward and backward passes of the convolution operation, projecting feature maps to a higher-dimensional spatial space [94, 95, 96]. In the deconvolution, the spatial dimension of an output tensor is increased, thus the output boundary is sometimes cropped to get a desired size.

In order to learn a hierarchical set of features, several shallow CAEs can be stacked into a multi-layer convolutional auto-encoder. The encoder consists of multiple stack of convolution, activation function, pooling layers and the decoder mirrors the encoder with unpooling, deconvolutional and activation function layers.

3D Convolution Autoencoders

2D convolutional neural networks lose temporal information of the input right after the first convolution layer. As can be seen in Figure 2.13 (a), although multiple frames are stacked to form an input, a 2D convolution layer treats a temporal dimension as input's channel, temporal information is collapsed completely. 3D convolutional networks have been proposed to learn spatio-temporal features for human action recognition in video [9, 97].



(a) 2D convolution on multiple frames (b) 3D convolution ($d < L$)

Figure 2.13: 2D and 3D convolution operations [9].

According to Tran et al. [9], a 3D convolutional network can learn effective video features with four properties: generic, compact, efficient and simple. A 3D convolution preserves the temporal information of the input resulting in an output tensor. Figure 2.13 (b) illustrates a 3D convolution which convolves a filter kernel size of $k \times k \times d$ with an input tensor size of $H \times W \times L \times C$, where k is kernel spatial size, d is kernel temporal depth, H is the height of the frame, W is the width of the frame, L and C are the length in number of frames and the number of channels ($C = 1$ in the illustration in Figure 2.13 (b)).

Beside the use of 3D convolution, a 2D pooling has been extended to 3D pooling [9], which applies max-pooling to the input data within a 3D pooling cube. Similar to a multi-layer convolutional autoencoder, a multi-layer 3D convolutional autoencoder consists of multiple stacks of 3D convolution, activation function, 3D pooling in the encoder and the decoder mirrors the encoder.

Long Short Term Memory Auto-encoders

Long Short-Term Memory (LSTM) units [98] can be used as an alternative to 3D convolution to learn changes in temporal dimension. An LSTM unit uses a memory cell to store information which is better at finding and exploiting a long range context. LSTMs have been used successfully to perform various supervised sequence learning tasks, such as speech generation [99], machine translation [100] and to generate image captions [101]. Moreover, they have applied on videos for visual object tracking [102], pedestrian trajectory prediction [103], visual recognition and description [41]. It has also been used to learn representations of video sequences [43].

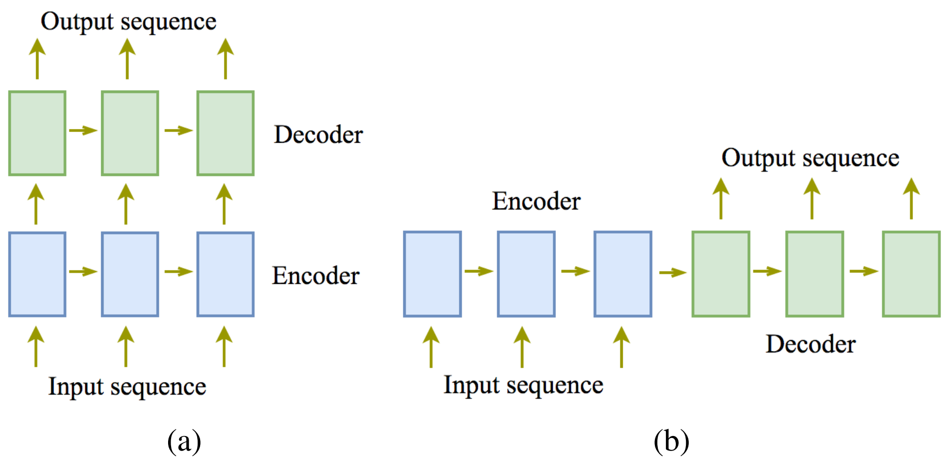


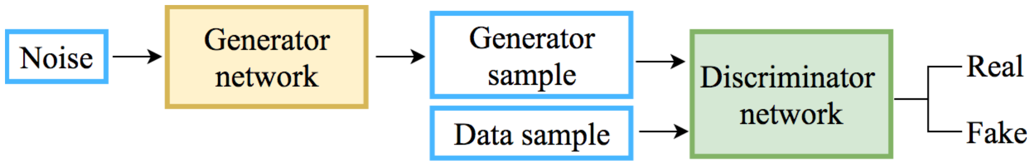
Figure 2.14: Two examples of a LSTM autoencoder.

LSTM unit can be used as an encoder and decoder in the network to extend the properties of AEs to model sequential data. Figure 2.14 shows examples of LSTM AEs. An LSTM encoder reads and compresses a sequence of frames into representations, from which the decoder attempts to produce a target sequence. LSTM units in the autoencoder can be a fully connected LSTM [43] or a convolutional LSTM [104].

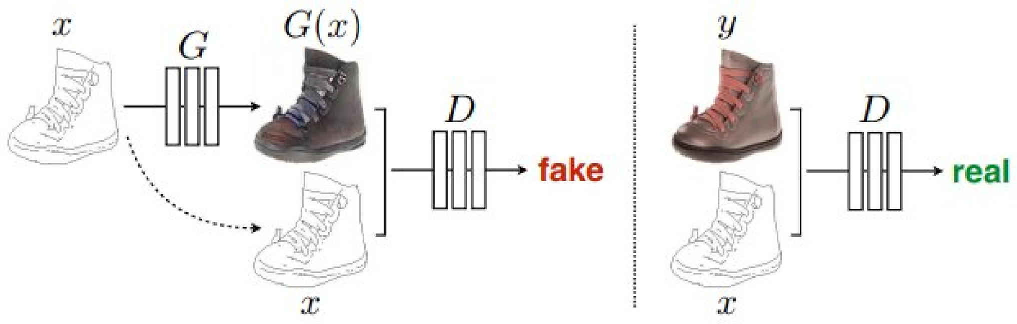
Generative Adversarial Networks

Other deep networks that can be trained on unlabelled data are Generative Adversarial Networks (GANs) [105], which have been recently proposed to generate data (e.g, images, video). The supervisory information in GANs is indirectly provided by an adversarial game between two independent networks: a generator G and a discriminator D. The generator G is trained to produce output images as real as possible that can fool D. The discriminator D is trained as well as possible, to distinguish between the real and the generated image. They both are optimised until the generator generates such realistic images that cannot be distinguished by the discriminator.

Based on the original GAN, Isola et al. [10] proposed to use conditional GANs for an image-to-image translation that synthesized photos from label maps or mapped edges to photos. Figure 2.15 shows the original GAN and a conditional GAN used for mapping edges to photos. GANs are trained to learn a mapping from random noise vector to output image. In contrast, the conditional GANs learn a mapping from observed image and random noise vector to output image. Both generator and discriminator of a conditional GAN observe an input image. The generator, which is a convolutional encoder-decoder, takes an input image and generate a new image. The noise distribution is provided in the form of dropout which is applied on several layers of the generator. The discriminator stacks the observed images with the real or generated images as its inputs. Instead of classifying an input as being real or fake, the discriminator classifies each patch in its input as real or fake, for the purpose of capturing local statistics. Besides optimising the GAN objective cost, the reconstruction error of the encoder-decoder is also considered.



(a) A Generative Adversarial Network



(b) A Conditional Generative Adversarial Network

Figure 2.15: (a) A Generative Adversarial Network and (b) an example of a conditional GAN for mapping edges to photos (Figure from [10]).

The above autoencoders can be employed for anomaly detection in two different ways. Firstly, the autoencoder can be trained on training data and then the encoder part can be used as a feature extractor to extract feature representations for both training and testing data. We review this method in Section 2.3.2. Secondly, the autoencoder can be trained end-to-end and the error can be used to detect anomalies. This method is reviewed in Section 2.3.3. Besides the use of an autoencoder, pre-trained deep models can be used to extract useful features for anomaly detection, which is represented in Section 2.3.2.

2.3.2 Deep feature learning for anomaly detection

Attracted by the capability of CNNs to produce such a generic representation, pretrained deep networks have been adopted for anomaly detection [11]. Sabokrou et al. [11] used the first three layers of the AlexNet model as a fixed feature extractor, then trained a third convolutional layer on top of these features using a sparse auto-encoder. Their architecture is shown in Figure 2.16. Each spatial point in the output tensors is considered as a feature representation for a corresponding region in an input image. To model normalities, two Gaussian models (G_1 and G_2) are trained on the feature representations of training data which are output tensors of the second and the third convolutional layers, respectively.

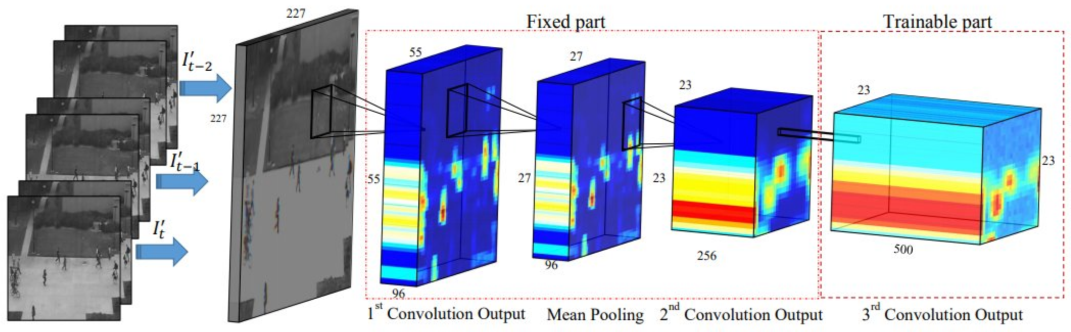


Figure 2.16: The fully convolutional network with a trainable layer on top of pretrained layers. (Figure from [11]).

The Mahalanobis distance from the feature vector and the Gaussian model is used to detect an anomaly. For test regions, the Gaussian model G_1 is used on feature vectors extracted from the second convolutional layer to classify regions as normal, suspicious or abnormal regions. A region that is an outlier from G_1 is considered to be abnormal. Those which are inliers but with a distance above a threshold are regarded as suspicious. For the suspicious regions, output tensors of the third convolutional layer are computed and used as their new representations. These features are then fed to the second Gaussian model G_2 to classify these suspicious regions into normal or abnormal regions. Results from G_1 and G_2 are combined to localize anomalies in the feature space which is then mapped to the original image.

The method shows that the pretrained networks can be used effectively for anomaly detection. However, such image based deep features are not directly suitable for videos due to the lack of motion modelling.

To overcome the above problem, Xu et al. [34] proposed a framework (Figure 2.17) that trains two separate stacked denoising autoencoders (SDAEs) to learn feature representations for both appearance in still image patches and motion patterns represented with optical flow. To learn the correlation between appearance and motion, image pixels and their corresponding optical flows are stacked to form input data for the third SDAE. After the SDAE-based features have been learned, multiple one-class SVM classifiers are trained on top of these features to model normalities and create anomaly maps. Finally, the three

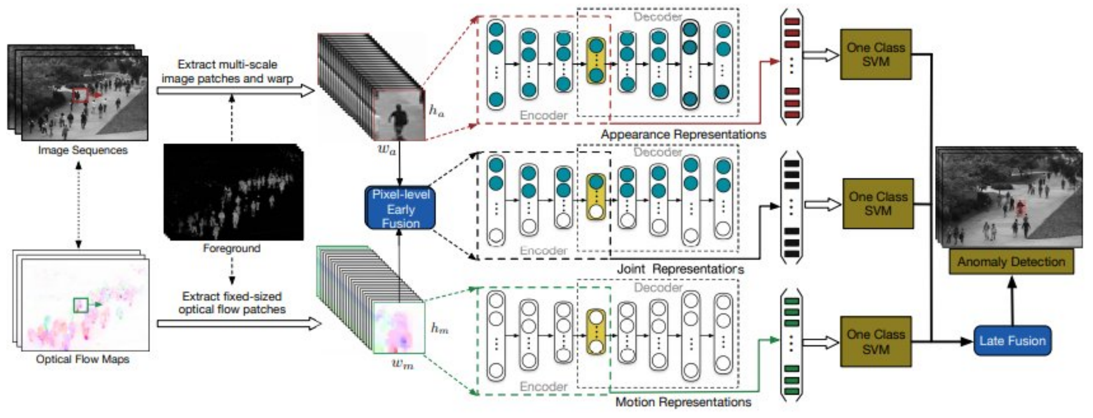


Figure 2.17: The overview of frameworks using three stacked denoising autoencoders to learn appearance, motion and joint representations. (Figure from [11]).

anomaly maps are fused to form a final map which is then thresholded to detect and localize anomalies.

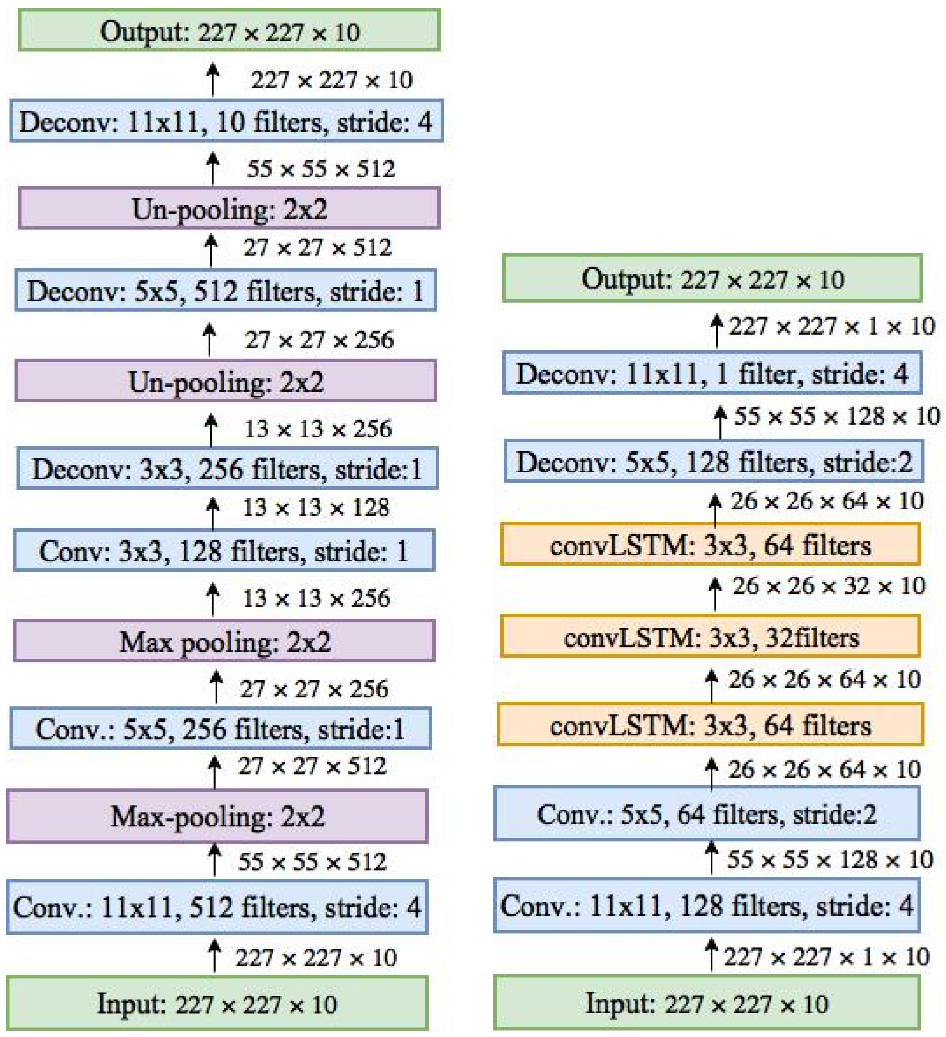
Similar to the method that uses pre-trained models, a model of normality needs to be trained on the learned features using another machine learning method (one-class SVM or Gaussian model, for example). In the next section, we review methods using end-to-end trainable deep networks for anomaly detection.

2.3.3 End-to-end deep network

Many methods using deep autoencoders on different inputs have been proposed for anomaly detection. Examples of input have been volumes of consecutive frames [12, 13, 14], 2D or 3D patches [12] or a combination of raw images and optical flow [12]. Sharing the same intuition with sparse coding methods, the autoencoder is trained to reconstruct normal motions and appearances with a low cost. Thus, it should derive higher reconstruction cost for abnormal motions and appearances. In this section, we review methods that use a convolutional autoencoder, a convolutional Long Short Term Memory (LSTM) autoencoder, a 3D convolutional autoencoder and a generative model (such as Generative Adversarial Networks) for anomaly detection.

A deep convolutional auto-encoder was trained by Hasan et al. [12] to learn discriminative regular patterns by reconstructing sequences of video frames from the training data. Their network architecture is shown in Figure 2.18 (a).

The network is trained to minimize an objective function with Euclidean loss and L_2



(a) Convolutional auto-encoder (b) Spatial-temporal auto-encoder.

Figure 2.18: Stacked convolutional auto-encoders used for anomaly detection. (a) Spatial-temporal information is learned on a sequence of 10 frames with a convolutional auto-encoder (Figure reproduced from [12]) and (b) a convolutional LSTM is applied on top of convolutional layer's feature maps to learn temporal information (Figure reproduced from [13]).

regularization as follows:

$$\mathcal{L} = \frac{1}{2N}\sum_i ||\mathbf{X}_i - f_{\mathbf{W}}(\mathbf{X}_i)||_2^2 + \gamma||\mathbf{W}||_2^2 \tag{2.8}$$

where $\mathbf{X}_i$ is the i^{th} volume, N is the batch size, γ is a hyper-parameter to balance the loss and the regularization; $f_{\mathbf{W}}$ is a convolutional auto-encoder with its weights $\mathbf{W}$.

After training a model, reconstruction error of a testing video volume is used to derive a regularity score which is then used to detect a volume as normal or abnormal. The method shows competitive results on five datasets. However, 2D convolutional and pooling layers only perform spatially. Though consecutive frames are stacked in input volumes, the temporal information may be lost after the first convolutional layer.

Inspired by the ability of a convolutional LSTM [104] to learn temporal dynamics and its potential for anomaly detection [106], Chong and Tay [13] combined convolutional LSTM layers with a convolutional auto-encoder to learn both spatial and temporal information for anomaly detection. Their network architecture is shown in Figure 2.18 (b). The key innovation in this work is the use of a convolutional encoder to learn high-level, compressed representations. Then convolutional LSTM layers encode temporal information on top of these feature activations. However, this model only learns temporal dynamics on high-level representations.

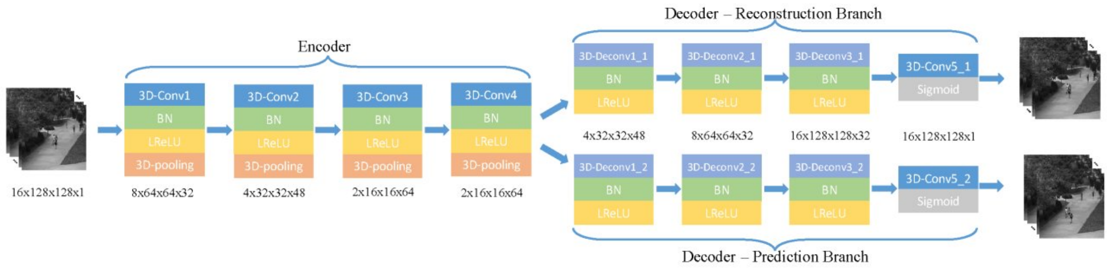


Figure 2.19: Architecture of a 3D convolutional auto-encoder for anomaly detection with reconstruction and prediction branches. Figure from [14].

Zhao et al. [14] used a 3D convolutional autoencoder to learn regular spatio-temporal patterns. Sharing the same intuition with previous works in [12, 13], the 3D convolutional auto-encoder, trained on the normal data, should reconstruct abnormal motions and appearances with a high reconstruction error. Their network architecture is illustrated in Figure 2.19. In order to optimise the network, a predictor branch is added on the final encoding output tensor alongside with a reconstruction branch. The composite of two branches forces the encoding tensors not just to memorize the whole input but also to capture information about which objects/background are present and their motions so that the next motion can be predicted. The predictor can help the model to capture the trajectory of moving objects and forces the encoder to learn the temporal dynamics [14].

After training the model with reconstruction error and prediction error, only reconstruction error is used to derive a regularity score as in previous works [12, 13]. The 3D convolutional autoencoder learns longer temporal regular dynamics since more video frames are stacked into the input volumes.

Ravanbakhsh et al. [53] adapted the image-to-image translation framework by Isola et al. [10] to solve the anomaly detection problem. The authors trained two conditional GANs: one network mapped raw images to optical flow fields and another network mapped optical flows to raw images. The models are trained on the normal data only. Thus the generator learns how to generate the normal pattern while the discriminator learns how to distinguish the generated sample and the real normal sample. After training the model, both generator and discriminator can be used for anomaly detection. The reconstruction loss from the generator is used the same way as in the previous works [12, 13, 14]. Moreover, the discriminator is trained to discriminate real and fake patches on normal data, it should produce high probability for a real normal patch. Moreover, the discriminator has never observed abnormal data during training, anomalies may be seen as fake samples, thus it produce low probability for anomalies.

In order to compare the performance, the above methods have been evaluated on some challenging anomaly detection datasets. In the next section, we review datasets and evaluation measures that are commonly used. We also use these datasets and these measures to evaluate our methods and to compare with state of the art results.

2.4 Experimental validation

2.4.1 Evaluation measure

In this section, we present three commonly used measures for anomaly detection: frame-level [17], pixel-level [17], and object-level [16]. An algorithm classifies frames into those that contain an anomaly and those that do not. These measures are obtained by comparing the detection results and the ground-truth. In the ground truth, a frame containing an anomaly is denoted as a "positive" frame while a normal frame is a "negative" frame.

The true and false positives under the three criteria are:

- Frame-level. If a frame contains at least one abnormal pixel, it is considered as abnormal. If the corresponding ground-truth is abnormal, it is a true positive. Otherwise, it is a false positive. The frame-level criterion does not verify the locations where the actual anomalies appear. It is therefore possible for some true positives obtained by the lucky occurrences of detection errors and true abnormalities. The pixel-level criterion can solve this problem.

- Pixel-level. One frame is a true positive if its ground-truth map of anomalous pixels is more than 40% covered by a map of predicted anomalous pixels. A frame is a false positive if it is negative and any of its pixels are predicted as abnormal. This measurement is better than frame-level, reflects the accuracy of anomaly location. However, it only focuses on the overlap between the detection and the ground-truth but ignores the union between them. A pixel level is satisfied even when there are many false positives. For example, when the all pixels in a frame are detected as anomalies, more than 40% of the true abnormal pixels are covered, thus that frame is labelled abnormal. However, it is not a good detection since it contains a lot of false positive pixels.

- Object-level. Object level measurement is the evaluation method for object detection in which the overlap is defined as the following:

$$\text{Overlap} = \frac{\text{Area}(\text{detection} \cap \text{groundTruth})}{\text{Area}(\text{detection} \cup \text{groundTruth})} \tag{2.9}$$

  A detection is true positive when $\text{Overlap} > v$ where v is a predefined threshold.

For frame-level and pixel level, a receiver operating characteristic (ROC) curve is employed to evaluate the performance. The ROC curve is the curve of True Positive Rate (TPR) versus False Positive Rate (FPR), generated by varying an acceptance threshold (Figure 2.20). TPR and FPR are defined as in Equation 2.10 where True/False Positive/Negative are counts for the corresponding class.

$$\begin{aligned} \text{TPR} &= \frac{\text{True Positive}}{\text{True Positive + False Negative}} \\ \text{FPR} &= \frac{\text{False Positive}}{\text{False Positive + True Negative}} \end{aligned} \tag{2.10}$$

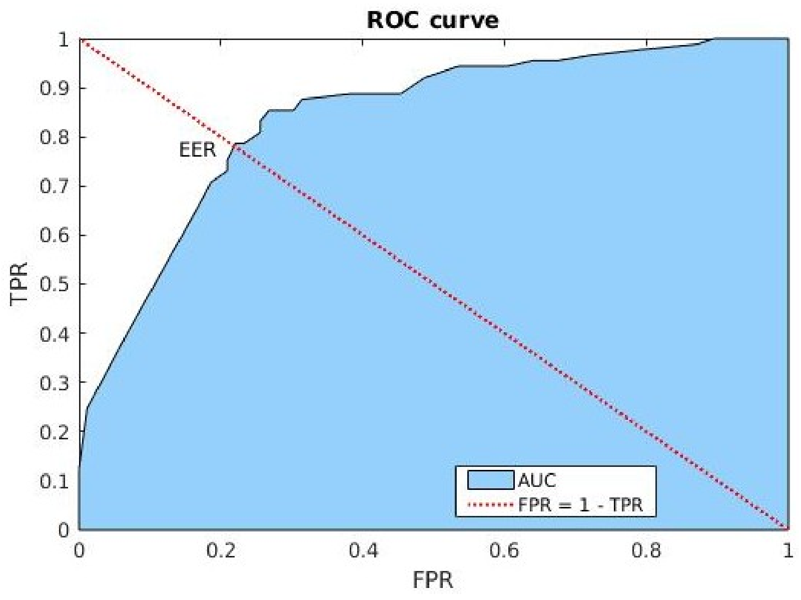


Figure 2.20: The ROC curve, where the blue area is AUC and the intersection between the line (FPR = 1 - TPR) and the curve is EER. A better method gives higher AUC and lower EER.

Performance is then evaluated using equal error rate (EER) which is the ratio of misclassified frames at which EER $=$ FPR $= 1 -$ TPR, and the area under the ROC curve (AUC). A better method gives higher AUC and lower EER.

For object-level, the accuracy is employed:

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{True Positive + True Negative + False Positive + False Negative}} \tag{2.11}$$

2.4.2 Datasets

There are many datasets available for anomaly detection. In this section, we review three commonly used datasets that we also use to conduct experiments and compare performance: UCSD [17], Avenue CUHK [16] and Subway Entrance/Exit [107].

UCSD anomaly detection dataset

The UCSD dataset [17] was acquired with a stationary camera mounted at an elevation, overlooking pedestrian walkways. The dataset contains different crowd densities, ranging from sparse to very crowded. In the normal setting, the video contains only pedestrians and the anomalous patterns are the presence of non-pedestrians on a walkway (bicyclists, skaters, small carts, and people in wheelchairs). In some frames the anomalies occur at

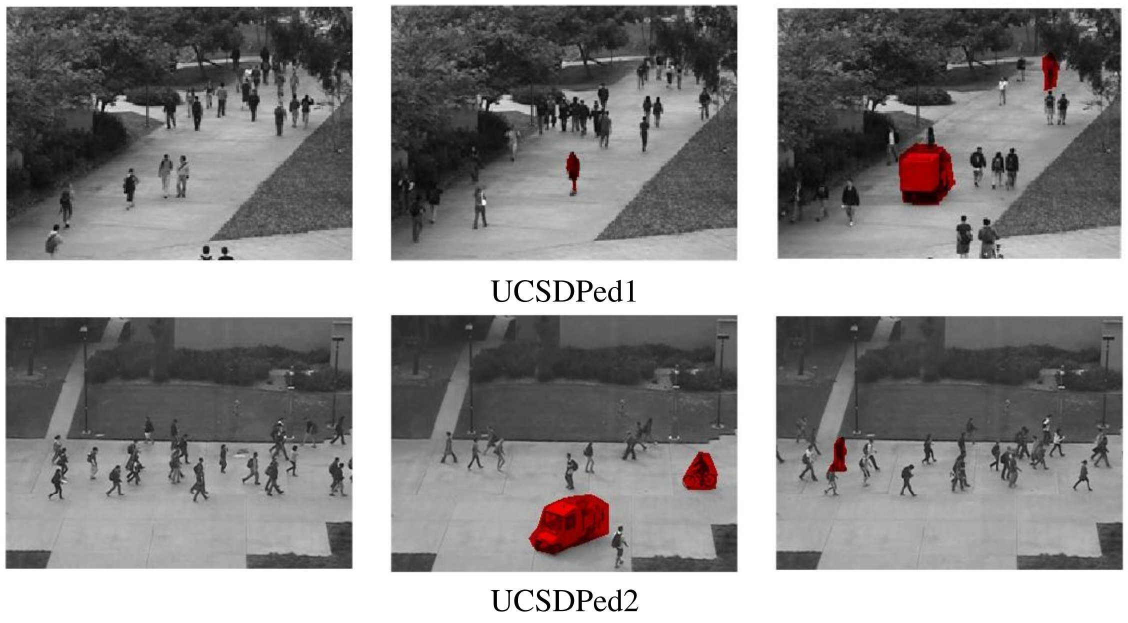


Figure 2.21: Sample normal/abnormal frames in UCSDPed1 (top row) and UCSDPed2 (bottom row). Anomalous pixels are shown in red.

multiple locations. The data was split into two subsets as shown in Figure 2.21, each corresponding to a different scene. The first one, denoted as UCSDPed1, contains clips of 158×238 pixels and depicts a scene where groups of people walk toward and away from the camera. This subset contains 34 normal video clips (6,800 frames) for training and 36 testing video clips (7,200 frames) containing one or more anomalies for testing. 36 testing clips are annotated with frame-level ground-truth while only 10 clips are annotated with pixel-level ground-truth. The second, denoted as UCSDPed2, has spatial resolution of 240×360 and contains scenes with pedestrian movement parallel to the camera plane. This contains 16 normal video clips (2,550 frames) for training, together with 12 test video clips (2,010 frames). The whole testing clips of UCSDPed2 has both frame-level and pixel level annotations. The composition of abnormal events in each subset is illustrated in Table 2.1 in which some testing clips contain more than one abnormal types and instances.

In general, UCSDPed1 is more challenging than UCSDPed2 as the angle of camera results in larger perspective distortion and more significant scale variation of moving objects. Furthermore, abnormal events in Ped1 include not only motion anomalies caused by bikers, skaters and small carts, but also contextual anomalies (e.g., pedestrian walking across the grass).

Table 2.1: Composition of abnormal events in the UCSD dataset.

Scene	# Normal clips	# Abnormal clips/abnomal instances					
		Biker	Skater	Cart	Walk across	Other	Total
UCSDPed1	34	19/28	13/13	6/6	3/4	3/3	36/54
UCSDPed2	16	11/19	3/3	1/1	0/0	0/0	12/23

Avenue (CUHK) dataset

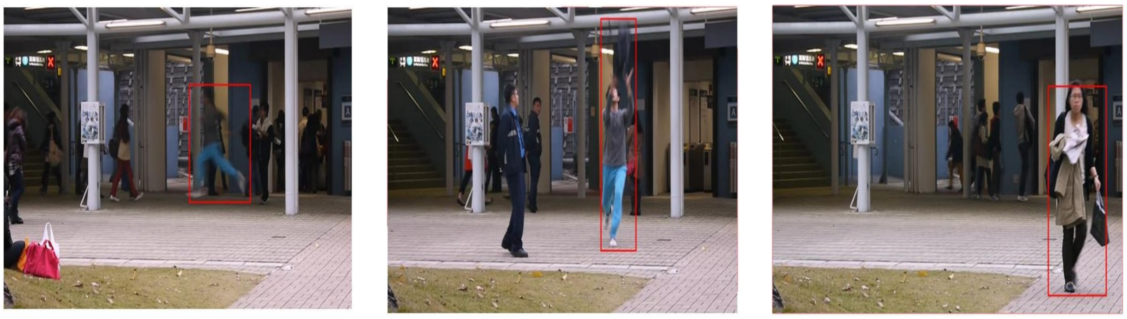


Figure 2.22: Examples of abnormal frames in Avenue dataset, where red boxes correspond to abnormal events.

Table 2.2: Groundtruth of Avenue dataset.

	Run	Loiter	Throw	Opposite direction	Total
Ground truth	12	8	19	8	47

The Avenue dataset [16] contains 16 training and 21 testing video clips with 47 abnormal events in total as summarized in Table 2.2. The videos were captured on the CUHK campus avenue with 30,652 (15,328 training, 15,324 testing) frames in total. Resolution of each frame is 360×640 pixels. The dataset contains some challenges: a few outliers are included in the training data, some normal patterns seldom appear in training data and the camera is shaken slightly in some testing frames. Locations of anomalies are marked in pixel-level ground-truth annotations for all frames in testing videos. Figure 3.14 illustrates some abnormal frames which contain running, throwing and someone moving in the opposite direction to normal.

Subway dataset

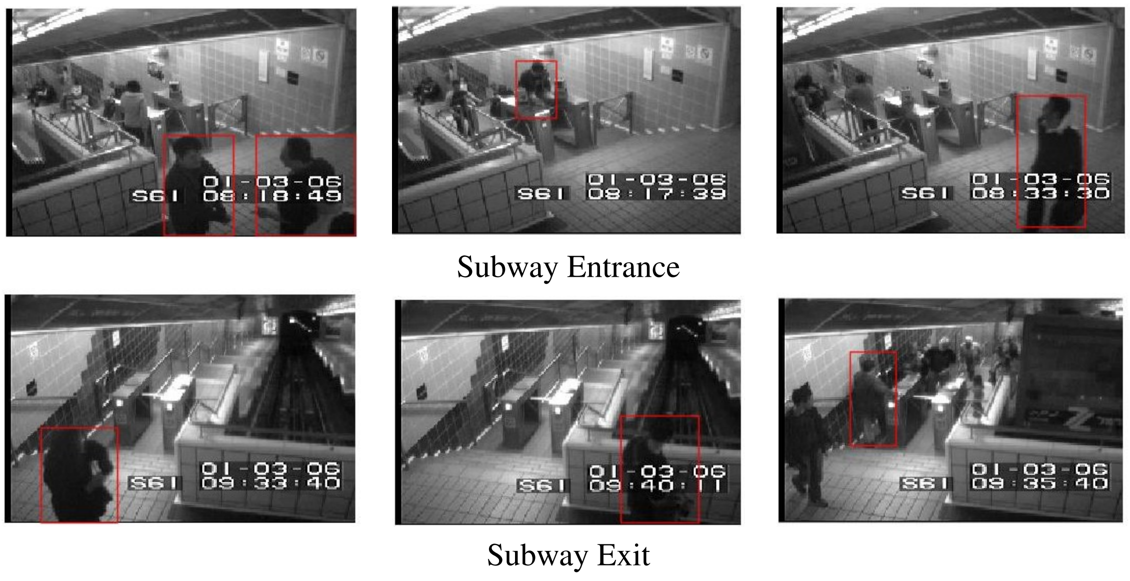


Figure 2.23: Examples of anomalous frames in Subway Entrance (top row) and Subway Exit dataset (bottom row). Red boxes correspond to abnormal events.

In the subway surveillance dataset [107], video sequences were taken from surveillance camera at a subway station, with one camera at the exit and a second at the entrance. In both videos, there are roughly 10 people walking around in a typical frame, with a frame size of 512×384. The details of each subset are as follows:

- Subway Entrance: is 1 hour 36 minutes long, with 144,249 frames. Normal frames in the first 12 minutes are used for training a model.
- Subway Exit: is 43 minutes long with 64,901 frames in total. Video frames in the first 5 minutes are used for training a model.

The number of abnormal events of both Subway Entrance and Subway Exit are defined in Table 2.3. These anomalies are "wrong direction", "loitering", "no payment", "irregular interaction" and "misc" which refers to miscellaneous, including suddenly stop, running fast or cleaning the wall.

Table 2.3: Groundtruth of Subway dataset.

	Wrong direction	Loiter	No payment	Irregular interaction	Misc	Total
Exit	9	3	0	0	7	19
Entrance	26	14	13	4	9	66

2.5 Summary

In this chapter we presented related work in the areas of anomaly detection using hand-crafted feature descriptors and classical machine learning methods. Moreover, methods based on deep learning were also reviewed. In each case, we described the strengths and limitations of the related approaches. Here we present a conclusion on how the presented related work affected the design of our two main frameworks.

The related work on feature descriptors showed that motion based features are crucial in most of the state of art methods. However, all of these features are hand-crafted features. Based on the related works on autoencoders, motion feature representations can be learned through a stacked autoencoder.

We share the same motivation with a feature representation learning approach using an autoencoder in the literature of anomaly detection, however, the method used fully connected autoencoders that can not capture the spatial information. We use a convolutional autoencoder to learn motion features. Moreover, we apply a sparsity constraint on the convolutional autoencoder which has been shown to be effective on learning features for a classification task. We present our method in detail in Chapter 3.

A limitation of using autoencoders to form the feature extractor is the use of another classical machine learning method (such as, one class SVM or a Gaussian model) to learn the model of normality on features extracted from training data. The whole system cannot be trained end-to-end. We adopt a convolutional Long Short Term Memory (LSTM) encoder-decoder to learn an end-to-end trainable network for anomaly detection. This model is trained to work directly on raw image data and learn temporal dynamics from it. Although we share the same motivation with state-of-the-art methods, we learn the temporal dynamics on hierarchical spatial feature activations from low-level to high-level. Beside the use of reconstruction error, we adopt prediction error for anomaly detection.

Based on these concepts, in Chapter 3 we present our method that uses a convolutional winner-take-all autoencoder to learn motion feature representations and a one-class SVM for modelling normalities. Our end-to-end trainable architecture using a convolutional LSTM encoder-decoder is presented in Chapter 4. Our methods are evaluated with current evaluation measures and commonly used datasets.

Chapter 3

Convolutional Winner-Take-All Autoencoder for anomaly detection

3.1 Introduction

In this chapter, we propose a method for video anomaly detection using a convolutional Winner-Take-All (WTA) autoencoder that has recently been shown to give competitive classification results [108]. The method builds on state of the art approaches to anomaly detection using a convolutional autoencoder and a one-class SVM (OCSVM).

Autoencoders has been used for anomaly detection in previous works [12, 34]. Xu et al. [34] built deep networks based on stacked de-noising autoencoders to learn appearance, motion and joint representations. Then OCSVMs were used to learn a model of normality and to detect anomalies. Hasan et al. [12] trained a fully connected auto-encoder and a fully convolutional autoencoder. Then, the authors used a regularity score which was derived from a reconstruction error for anomaly detection.

In our work, we use a convolutional autoencoder to learn patterns in local flow features, but instead of applying across the whole field of view (FoV) [12], we apply within fixed-size windows (so-called patches). With smaller windows, we are able to use spatial WTA step to produce a sparse (and compressive) representation as in [108]. This sparse representation promotes the emergence of distinct flow-features during training. Similar to Xu et al. [34], we use an autoencoder on patches, coupled with a OCSVM for anomaly detection and

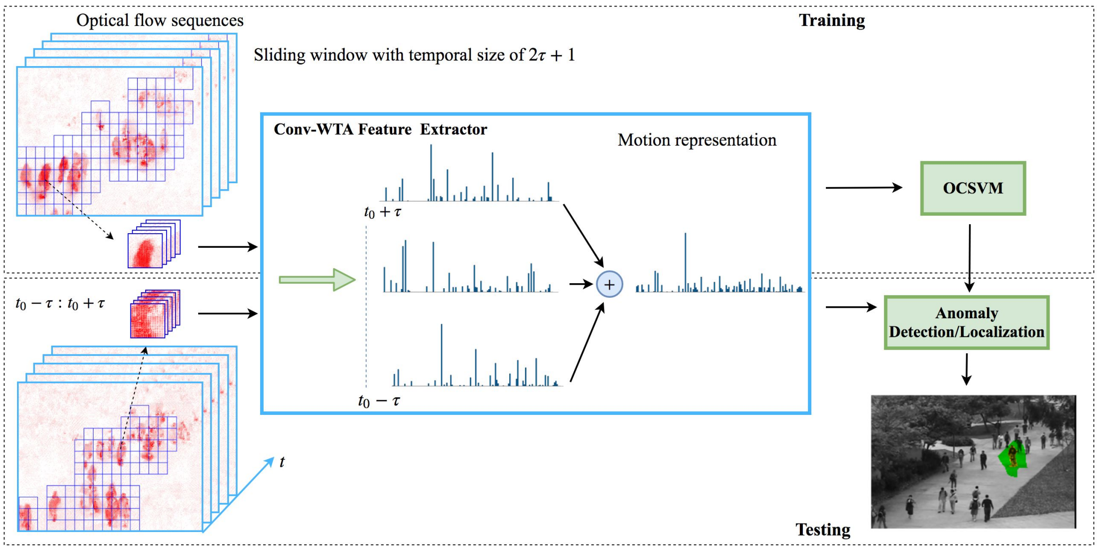

Optical flow sequences
Training
Sliding window with temporal size of $2\tau + 1$
Conv-WTA Feature Extractor
Motion representation
$t_0 + \tau$
$t_0 - \tau$
OCSVM
Anomaly Detection/Localization
$t_0 - \tau : t_0 + \tau$
t
Testing

Figure 3.1: Overview of the method using a spatial sparsity Convolutional Winner-Take-All autoencoder for anomaly detection.

localization. However, their autoencoder is fully connected and therefore learns larger flow features. By using a convolutional autoencoder within the window, coupled with a sparsity operator (WTA), we learn smaller shift-invariant, generic flow-features that are potentially more discriminative for the OCSVM. We also use local normality modelling in which the field of view is partitioned into regions and OCSVM is independently used within each region. Moreover, we only use optical flow data as input, instead of the combination of optical flow and appearance that has been used previously [12, 34].

The key novelties are (1) using the motion-feature encoding extracted from a convolutional autoencoder as input to a one-class SVM rather than exploiting reconstruction error of the convolutional autoencoder, and (2) introducing a spatial winner-take-all step after the final encoding layer during training to introduce a high degree of sparsity. We demonstrate an improvement in performance over the state of the art on UCSD and Avenue (CUHK) datasets.

Figure 3.1 shows the overview of our method using a spatial sparsity convolutional WTA autoencoder for anomaly detection. Motion representations are extracted for consecutive small patches which are then used to train a OCSVM (in training phase) or to detect anomalies (in testing phase).

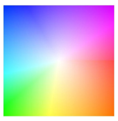
3.2 Our method

3.2.1 Extracting foreground patches

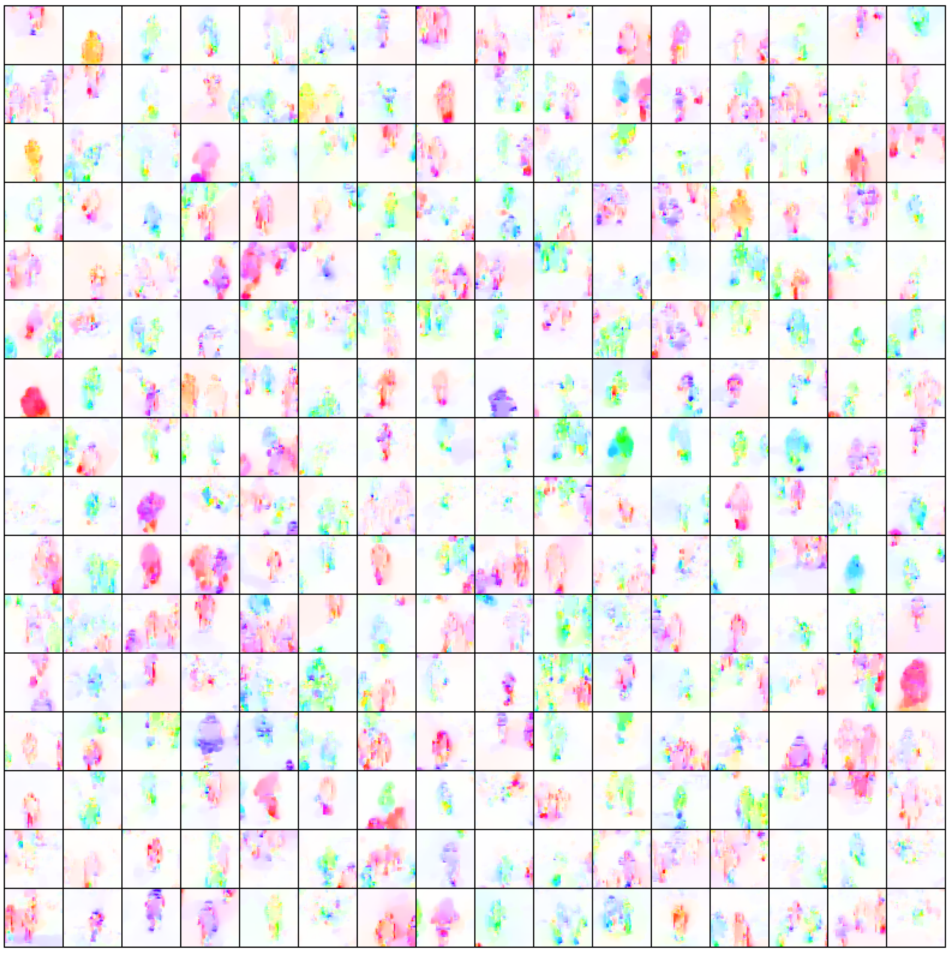
In common with recent approaches [2, 18, 33], we look for anomalies via dense optical flow fields computed from successive pairs of video frames presented by Liu [109]. We assume that anomalies will only be found where there is non-zero optical flow in the image plane. Thus, we do not attempt to detect anomalous appearances of static objects.

Two different patch sizes and methods are adopted to extract foreground patches for training an autoencoder and for anomaly detection. For training autoencoders, we observe that patches size of 48×48 can cover the whole object motions and the complex interactions between objects in all of our experimental datasets. Moreover, with this size, training is faster than training the autoencoder on an image size. Smaller size such as 24×24 are also considered however it takes longer time to obtain smooth and clean deconvolutional filters. Some samples of optical flow patches extracted for training autoencoders are shown in Figure 3.2. We extract them through three steps: firstly a foreground mask is constructed by thresholding the optical flow magnitude of each pixel with a fixed value (empirically set at 0.5 in our experiments). Secondly, connected components are found. Finally, from a center of each connected component, a patch is extracted.

For anomaly detection, patches are extracted by a moving window with 50% overlap. Those patches with the accumulated optical flow squared magnitude above a fixed threshold (empirically set at 10 in our experiments) are foregrounded for further processing; other patches are discarded. Figure 3.3 depicts the result of extracting foreground patches. This process is designed to eliminate most of the background, thereby reducing the computational cost of further processing. This method can be done by using convolution with a matrix of ones in which size of the matrix is the patch size.



(a)



(b)

Figure 3.2: (a) The flow field color coding [15] used in this chapter, where flow-vector angle and magnitude are represented by hue and saturation; (b) Examples of training patches.

3.2.2 Convolutional Winner-Take-All autoencoder

The Convolutional Winner-Take-All Autoencoder (Conv-WTA) [108] is a non-symmetric autoencoder that learns hierarchical sparse representations in an unsupervised fashion. The encoder typically consists of a stack of several ReLU convolutional layers with small filters and the decoder is a linear deconvolutional layer of larger size. A deep encoder with small filters incorporates more non-linearity and effectively regularises a larger filter (e.g 11×11)

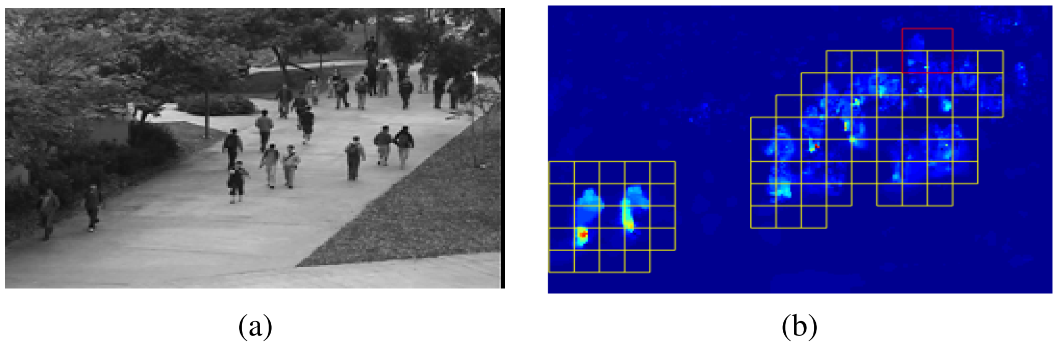


Figure 3.3: Foreground patches extraction using a sliding window and thresholding of the accumulated optical flow squared magnitude. (a) Video frame at time t. (b) Map of the flow magnitude (from frames t and $t+1$) with overlapping foreground patches superimposed; the red square delineates a single (24×24) foreground patch.

by expressing as a decomposition of smaller filters (e.g. 5×5) [82]. Like [108], we use an autoencoder with three encoding layers and a single decoding layer (Figure 3.4), giving a pipeline of tensors $H_l \times W_l \times C_l$, with the input layer being an input foreground patch $\mathbf{P}$ of optical flow vectors of size $H_0 \times W_0 \times C_0$, where $C_0 = 2$. Zero-padding is implemented in all convolutional layers, so that each feature map has the same spatial size as the input.

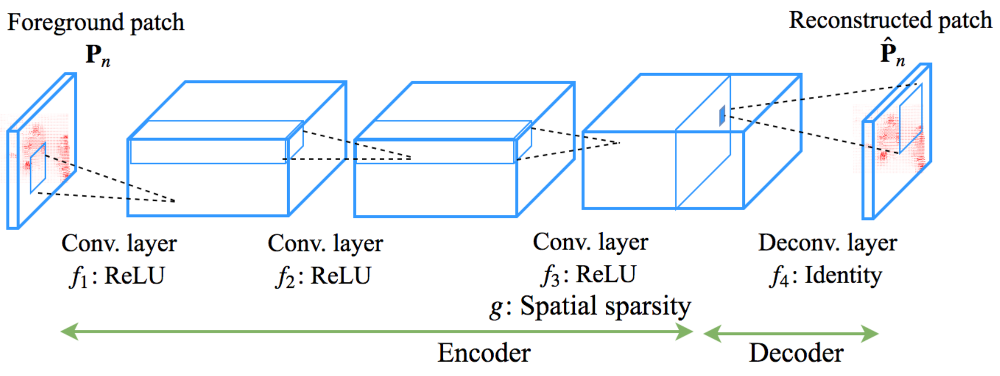


Figure 3.4: The architecture for a Conv-WTA autoencoder with spatial sparsity for learning motion representations.

Given a training set with N foreground patches ($\{\mathbf{P}_n\}_{n=1}^{N}$), the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ of each layer l are learnt by minimising the regularised least squares reconstruction error:

$$\frac{1}{2N}\sum_{n=1}^{N}\|\mathbf{P}_n - \hat{\mathbf{P}}_n\|_2^2 + \frac{\lambda}{2}\sum_{l=1}^{4}\|\mathbf{W}_l\|_F^2 \tag{3.1}$$

where $\|.\|_F$ denotes the Frobenius norm, and $\hat{\mathbf{P}}_n$ is the reconstruction of a patch $\mathbf{P}_n$. The regularization term λ is a hyper-parameter used to balance the importance of the reconstruction error and the weight regularization.

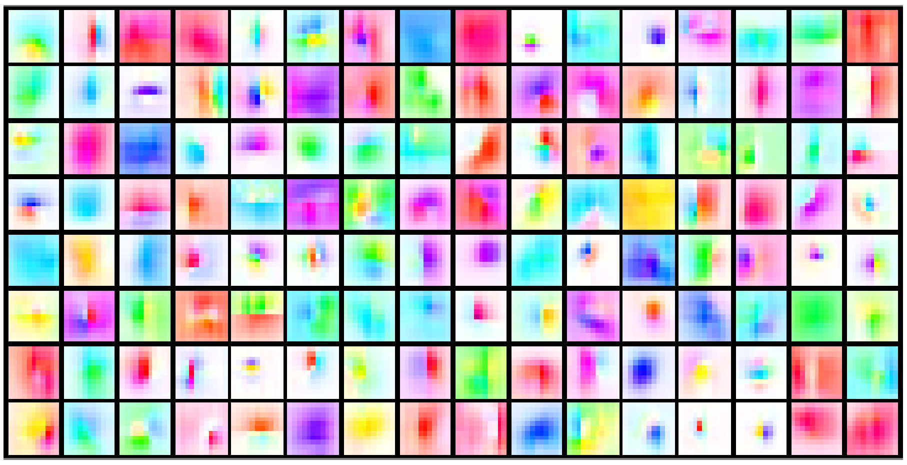
In the feedforward phase, after computing the encoding tensor $\mathbf{E}_3(x, y, c)$ (i.e. the output of f_3 in Figure 3.4), a spatial sparsity mapping is applied:

$$g(\mathbf{E}_3(x,y,c)) = \begin{cases} \mathbf{E}_3(x,y,c), & \text{if } \mathbf{E}_3(x,y,c) = \max_{x',y'}(\mathbf{E}_3(x',y',c)) \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

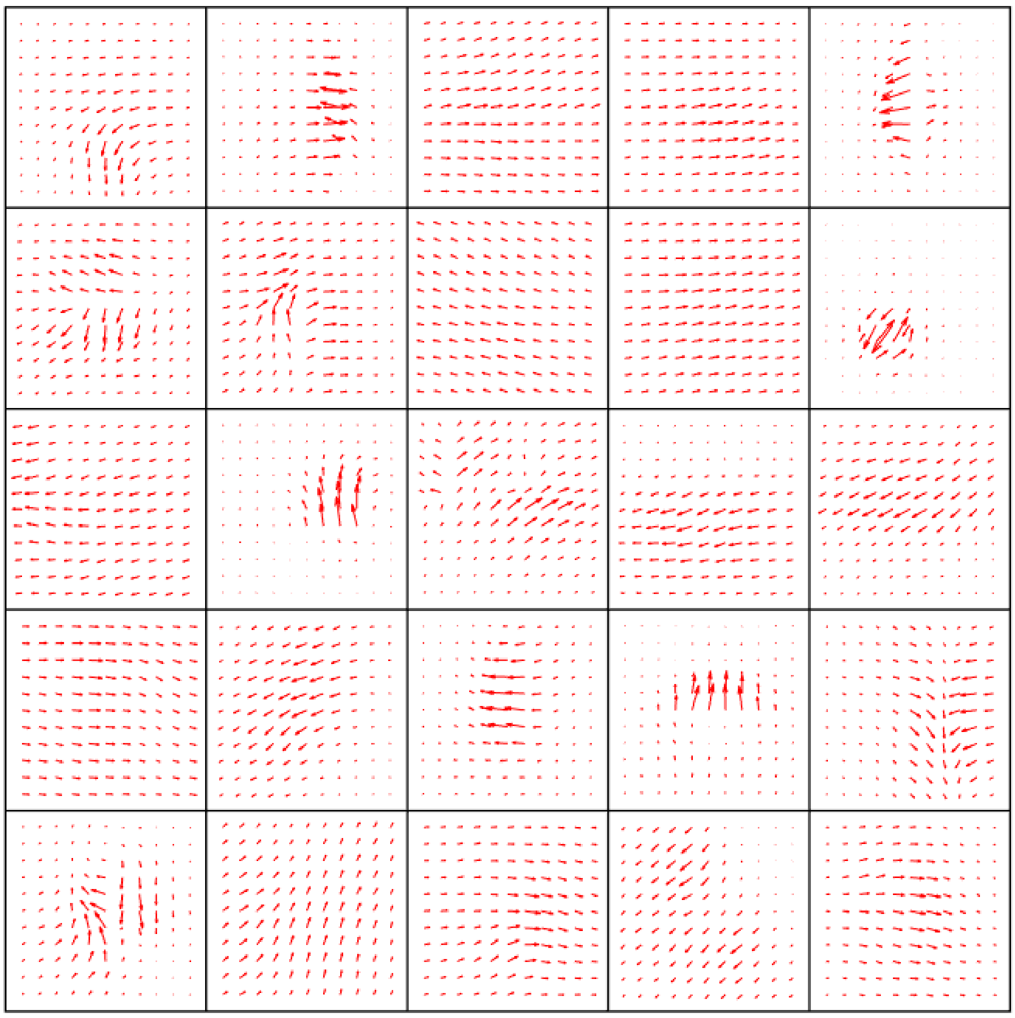
where (x, y, c) are the row, column and channel indices of an element in the tensor. The result $\mathbf{E}_4(x, y, c) = g(\mathbf{E}_3(x, y, c))$ has only one non-zero value for each channel. Thus, the level of sparsity is determined by the number of feature maps (C_3 for the third layer). Only the non-zero hidden units are used in back-propagating the error during training. We train the convolutional Winner-Take-All autoencoder on patches, then we use the encoder part with three convolutional layers (following by ReLU) to form a feature extractor. Figure 3.5 shows learned deconvolutional filters of the convolutional WTA autoencoder trained on patches extracted from UCSD database. We further describe how we combine the encoder part with a max pooling layer and a temporal average step to form a feature extractor in the next section.

3.2.3 Max pooling and temporal averaging for motion feature representation

After training the autoencoder, the encoder can be used as a feature extractor on patches. C_3 non-zero activations in the encoding tensor correspond to deconvolutional filters (shown in Figure 3.5) which contribute in the reconstruction of each optical flow patch. Using the full output tensor of size $H_3 \times W_3 \times C_3$ as a motion feature representation preserves all of the information, but is very large. Therefore, we extract a sparse and compressed motion feature representation by turning off spatial sparsity, removing zero padding in the three convolutional layers and applying max-pooling on the last ReLU feature maps, over the spatial region $p \times p$ with stride p. The max-pooling is only used following the training of the autoencoder with WTA. Thus we benefit from the sparse representation that WTA promotes, whilst still reducing the dimensionality of the coding so that it is tractable for the one class SVM. Crucially, WTA preserves the location of the maximum response in each filter, which is critical to successfully decoding and training the autoencoder to reduce reconstruction error. The location of the maximum response is less critical for anomaly detection and hence max-pooling, which greatly reduces dimensionality, is sufficient once training is complete.



(a)



(b)

Figure 3.5: Learned deconvolutional filters of the Conv-WTA autoencoder trained on the UCSDPed1 and UCSDPed2 optical flow foreground patches: (a) visualisation of 128 filters, and (b) displacement vector visualisation of 25 filters.

To stabilise the output for each $H_0 \times W_0 \times C_0$ foreground patch extracted at time t, we compute the motion feature representations at the same patch location over a temporal window $\{t-\tau : t+\tau\}$ and average the outputs. This gives a final smoothed motion feature representation as output for each input foreground patch. Therefore, our convolutional WTA feature extractor is considered as the combination of three convolutional layers of the trained encoder with a max-pooling layer and a temporal averaging layer, which is shown in Figure 3.6.

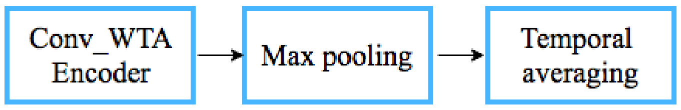


Figure 3.6: The convolutional WTA feature extractor.

To evaluate the effect of WTA operator in the convolutional autoencoder, we also train a convolutional autoencoder to learn feature representations. We describe the architecture of the autoencoder in the next section.

3.2.4 Convolutional autoencoder

In order to evaluate the efficiency of the WTA step in the autoencoder, this step is removed from the convolutional WTA autoencoder. The network becomes a traditional convolutional autoencoder with three convolutional layers in the encoder and one deconvoltional layer in the decoder. Using the same kernel sizes as the convolutional WTA autoencoder, output tensors are with high dimension. To compress the encoding tensor, a max-pooling layer is added on the top of the third convolutional layer. Figure 3.7 shows the architecture of the convolutional autoencoder with a max-pooling layer.

In a similar way to train the convolutional WTA autoencoder, we use the regularised least squares reconstruction error as in Equation 3.2 to train the convolutional autoencoder. Figure 3.8 compares training error of the convolutional WTA autoencoder and the convolutional autoencoder, the convolutional autoencoder converges to higher reconstruction error than the convolutional WTA autoencoder. A max-pooling layer loses the location information when compressing an encoding tensor, making a deconvolutional layer difficult to reconstruct. Figure 3.10 shows deconvolutional filters learned with the convolutional autoencoder.

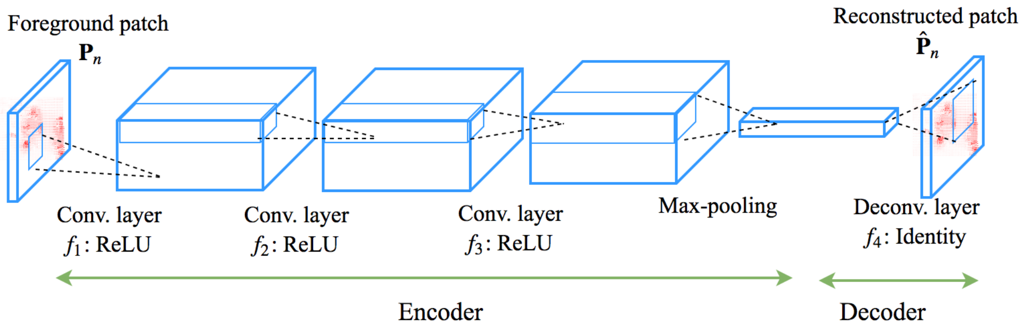


Figure 3.7: Convolutional autoencoder for learning motion representations.

After training, we keep the encoder with three convolutional layers and the max-pooling layer, adding a temporal averaging step on the top to form a convolutional feature extractor (Figure 3.8). This feature extractor has the same architecture as the convolutional WTA feature extractor (Figure 3.6), although the first three convolutional layers are trained in different ways (with a max-pooling layer or with a winner-take-all step).

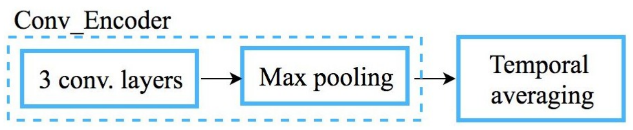


Figure 3.8: The convolutional feature extractor.

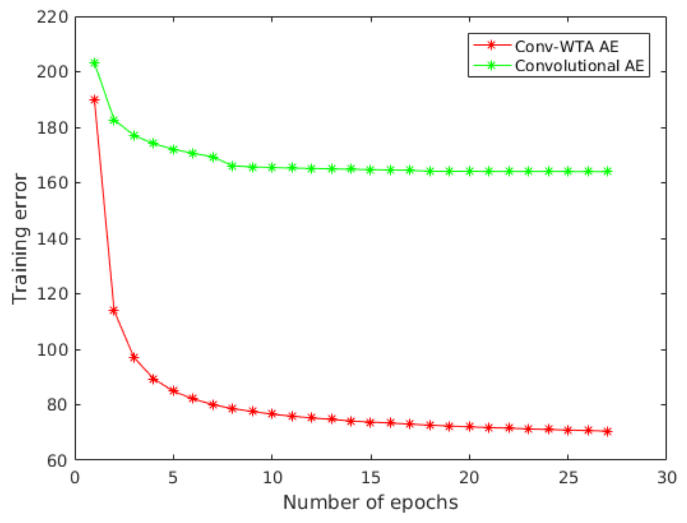
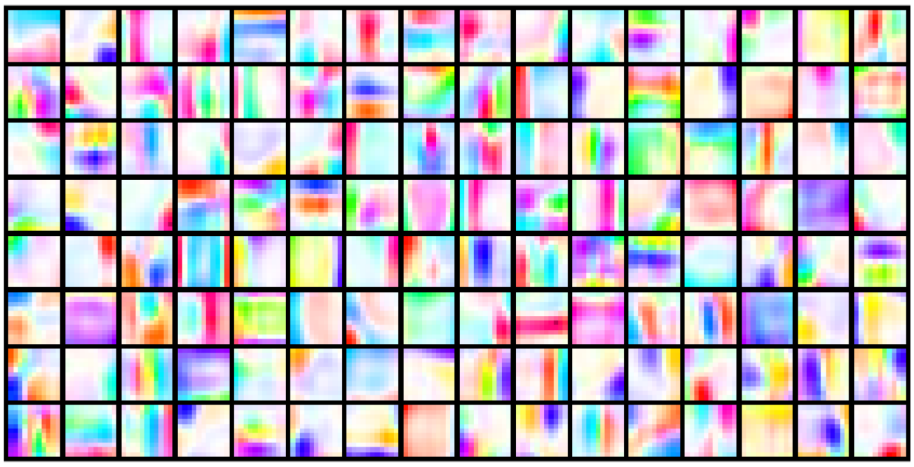
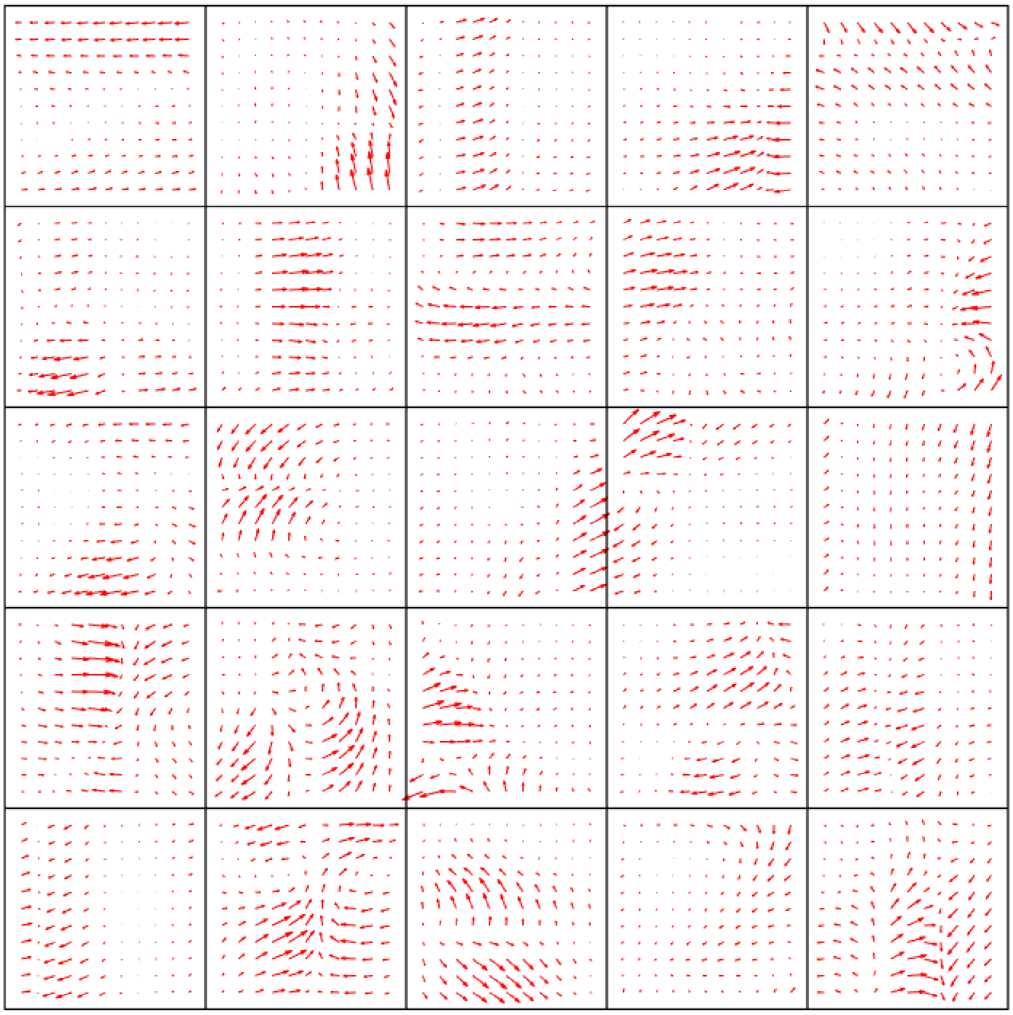


Figure 3.9: Training error of the convolutional autoencoder and the convolutional WTA autoencoder.



(a)



(b)

Figure 3.10: Learned deconvolutional filters of the convolutional autoencoder trained on the UCSDPed1 and UCSDPed2 optical flow foreground patches: (a) visualisation of 128 filters, and (b) displacement vector visualisation of 25 filters.

After extracting feature representations for patches in training data, these features are used to train a model of normality using one class SVM. One class SVM method is further described in the next section.

3.2.5 One class SVM modelling

One class SVM (OCSVM or unsupervised SVM) [76] is a widely used method for outlier detection. Given the final feature-based representations $\{\mathbf{d}_i\}_{i=1}^{M}$ for M normal foreground optical flow patches, we use OCSVM for learning a normality model. OCSVM aims to find an optimal hyperplane that separates the data points from the origin with maximum margin in the higher dimensional feature space:

$$f(\mathbf{d}) = \mathbf{w} \cdot \Phi(\mathbf{d}) - \rho \tag{3.3}$$

where Φ is a feature projection function that maps a feature vector $\mathbf{d}$ into a higher dimensional feature space, $\mathbf{w}$ is a decision hyperplane normal vector which is perpendicular to the hyperplane, and ρ is an intercept term. $\mathbf{w}$ and ρ are obtained by solving the following quadratic programming problem:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},\rho} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu M}\sum_i \xi_i - \rho \\ \text{subject to} \quad & \mathbf{w} \cdot \Phi(\mathbf{d}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0. \end{aligned} \tag{3.4}$$

the meta-parameter $\nu \in (0, 1]$ determines the upper bound on the fraction of outliers and the lower bound on the number of training examples used as support vectors; ξ_i are non-zero slack variables for penalizing the outliers.

By using multipliers $\alpha_i, \beta_i \geq 0$, a Lagrangian is introduced:

$$L(\mathbf{w}, \boldsymbol{\xi}, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu M}\sum_i \xi_i - \rho - \sum_i \alpha_i(\mathbf{w} \cdot \Phi(\mathbf{d}_i) - \rho + \xi_i) - \sum_i \beta_i \xi_i \tag{3.5}$$

Setting the derivatives with respect to $\mathbf{w}, \boldsymbol{\xi}, \rho$ equal to zero, $\mathbf{w}$ can be determined by $\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{d}_i)$ and the Equation 3.5 is optimized by solving dual form problem [77]. The decision function becomes:

$$f(\mathbf{d}) = \sum_i \alpha_i k(\mathbf{d}_i, \mathbf{d}) - \rho \tag{3.6}$$

where $k(\mathbf{d}_i, \mathbf{d}) = \Phi(\mathbf{d}_i) \cdot \Phi(\mathbf{d})$ is a kernel function. In our experiment, we employ a

Gaussian kernel:

$$k(\mathbf{d}_i, \mathbf{d}) = e^{-\gamma\|\mathbf{d}_i - \mathbf{d}\|^2} \tag{3.7}$$

When working with OCSVMs, feature normalization is commonly used as a preprocessing step. We use Min-Max scaling (or normalization) which refers to (independently) setting each dimension of the data to a fixed range of $[0, 1]$. We firstly compute the minimum value of each dimension across the training data and subtract this from each dimension. Then, each dimension is divided by its maximum value. We also include in Annex (Chapter 6) the comparison results of two different methods for normalization.

After training, given the optimal α_i, ρ as well as support vectors $\mathbf{d}_i$, an anomaly score for a representation $\mathbf{d}$ of a test patch can be estimated by computing $s(\mathbf{d}) = \rho - \sum_i \alpha_i k(\mathbf{d}_i, \mathbf{d})$ and then it is compared with a threshold thr to identify as a normal or an abnormal patch. In particularly, a test patch is abnormal when $s(\mathbf{d}) \geq thr$.

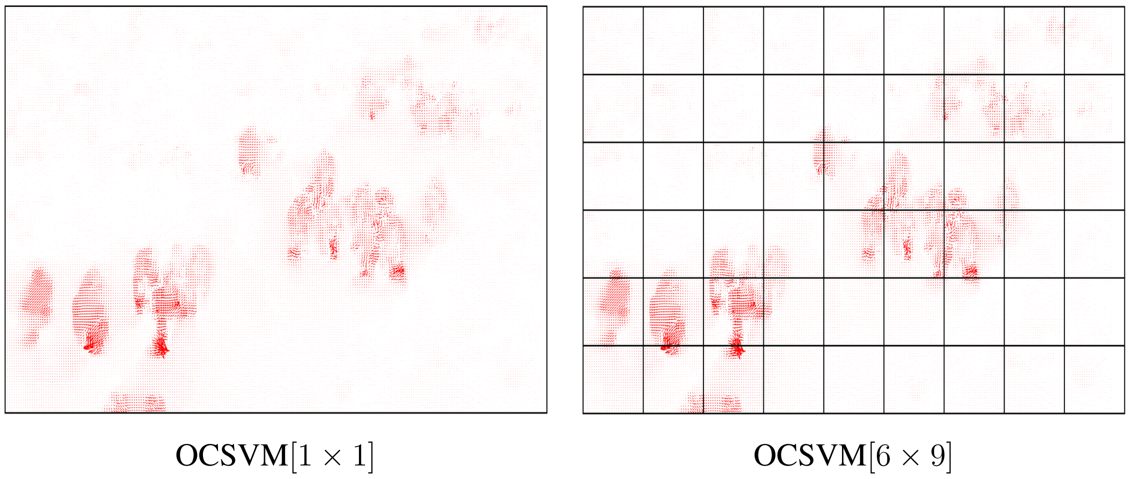


Figure 3.11: Examples of one-class SVM regions.

In order to capture variations in scale of flow patterns as object moves in depth through the scene, we divide the field of view into $I \times J$ regions. A separate OCSVM is learned from the foreground patches located in each region. Local modelling also helps to capture ‘contextual’ anomalies which are normal motions appearing at a specific part of the scene. For example, people walking on the grass surrounding a walkway may be a rare event, it should be therefore regarded as anomalous. Figure 3.11 shows examples of OCSVM regions used in the experiments, in which $I = J = 1$ for OCSVM$[1 \times 1]$ (so-called a global OCSVM) and $I = 6$, $J = 9$ for OCSVM$[6 \times 9]$.

3.3 Experimental evaluation

3.3.1 Dataset and Evaluation measures.

We use two datasets (UCSD and Avenue) in our evaluation. The UCSD dataset [17] contains two subsets of video clips, each corresponding to a different scene. The first one, denoted as UCSDPed1, contains clips of 158×238 pixels and depicts a scene where groups of people walk toward and away from the camera. This subset contains 34 normal video clips and 36 video clips containing one or more anomalies for testing. The second, denoted as UCSDPed2, has spatial resolution of 240×360 and contains scenes with pedestrian movement parallel to the camera plane. This contains 16 normal video clips, together with 12 test video clips. For our experiments on the UCSD dataset, we use ground-truth annotations from Mahadevan et al. [17]. The Avenue dataset [16] contains 16 training videos and 21 testing videos. In total there are 15,328 training frames and 15,324 testing frames, all with resolution 360×640.

We evaluate the method using the frame-level and pixel-level criteria proposed by Mahadevan et. all [17]. An algorithm classifies frames into those that contain an anomaly and those that do not. For both criteria, these predictions are compared with ground-truth to give the equal error rate (EER) and area under the curve (AUC) of the resulting ROC curve (TPR versus FPR) generated by varying an acceptance threshold. For a predicted anomalous frame to be correct, the pixel-level criterion [17] additionally requires that a ground-truth map of anomalous pixels is more than 40% covered by a map of predicted anomalous pixels. This criterion is well founded when the map of abnormal pixels is constrained to arise through thresholding a map of abnormality scores, as in [17]; otherwise it can be circumvented by setting every pixel in a frame as anomalous, when just one pixel is predicted to be anomalous - the frame-level score is not affected and the pixel-level criterion is always satisfied. In order to classify at the pixel level, bilinear interpolation was used with the computed patch scores.

3.3.2 Experimental Settings

Convolutional WTA autoencoder architecture and parameters

The Conv-WTA autoencoder architecture is 128conv5-128conv5-128conv5-128deconv11 with a stride of one, zero-padding of two in each convolutional layer and cropping of five in the deconvolutional layer. 128conv5 refers to the convolutional layer with 128 kernels of spatial size 5×5 and 128deconv11 refers to the deconvolutional layer with spatial kernel size 11×11. We train our model on 3×10^5 foreground optical flow patches of size 48×48 extracted from the UCSD dataset, using stochastic gradient descent with batch size $N_b = 100$, momentum of 0.9 and weight decay $\lambda = 5 \times 10^{-4}$ [37]. The weights in each layer are initialized from a zero-mean Gaussian distribution whose standard deviation is calculated from the number of input channels and the spatial filter size of the layer [87]. This is a robust initialization method that particularly considers the rectifier nonlinearities [87]. The biases are initialized to zero. A fixed value for the learning rate $\alpha = 10^{-4}$ is used following the first iteration. We use the MatConvNet toolbox [110], augmented to perform WTA.

We use the same architecture for the convolutional autoencoder, replacing a WTA step with a max-pooling layer.

One class SVM model

The LIBSVM library (version 3.22) [111] is employed for our experiments. The parameter ν is chosen from the range $\{2^{-12}, 2^{-11}, \ldots, 2^{0}\}$ and γ (in the Gaussian kernel) is from the range $\{2^{-12}, 2^{-11}, \ldots, 2^{12}\}$. Both parameters are selected by 10-fold cross validation on training data containing only normal activities.

For the UCSD dataset, we resize the frame resolution to 156×240. We evaluate performance with three subdivisions of the field of view: $[1 \times 1], [6 \times 9]$ and $[12 \times 18]$. The first of these is equivalent to operating over the entire field of view. For the Avenue dataset, we resize the frame resolution to 120×156 which is close to one scale used in [16]. Here we evaluate performance with three different subdivisions of the field of view: $[1 \times 1], [4 \times 6]$ and $[8 \times 12]$. In both cases, the sub-divisions are chosen to divide at pixel boundaries. 10-fold cross validation is used once on each dataset for the OCSVM$[1 \times 1]$ model to select values for the parameters to be used in all experiments ($\nu = 2^{-9}$ and $\gamma = 2^{-7}$).

3.3.3 Quantitative Analysis

3.3.3.1 Comparison with the state of the art

In this section, we compare the proposed framework with state of the art methods on the UCSD and Avenue datasets. Each method is compared on both ROC curves (Figure 3.12, 3.13 and 3.14) and the EER/AUC metric (Tables 3.1 and 3.2). State of the art results in Tables 3.1 and 3.2 are taken from published papers, some special cases are noted with ‘*’and are explained in corresponding tables.

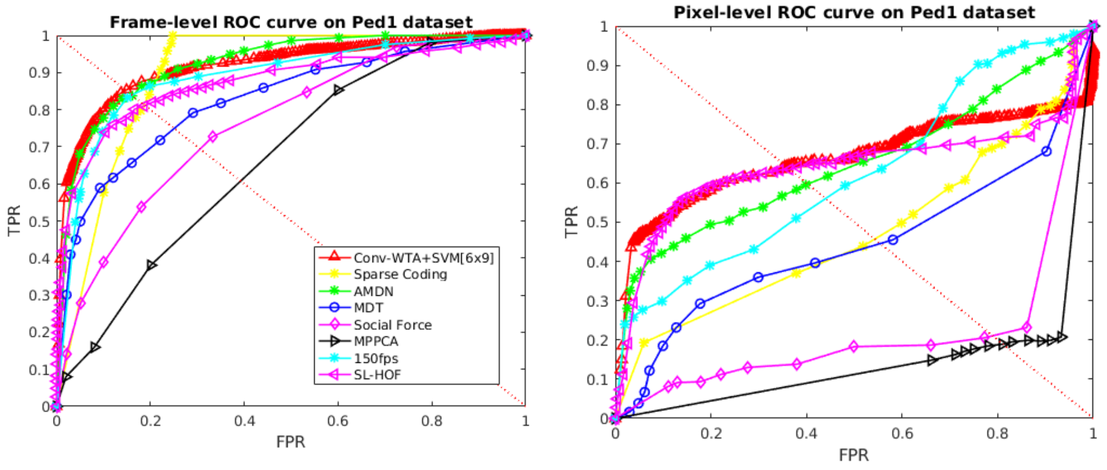


Figure 3.12: Frame-level and pixel-level evaluation on the UCSDPed1. The legend for the pixel-level (right) is the same as for the frame-level (left).

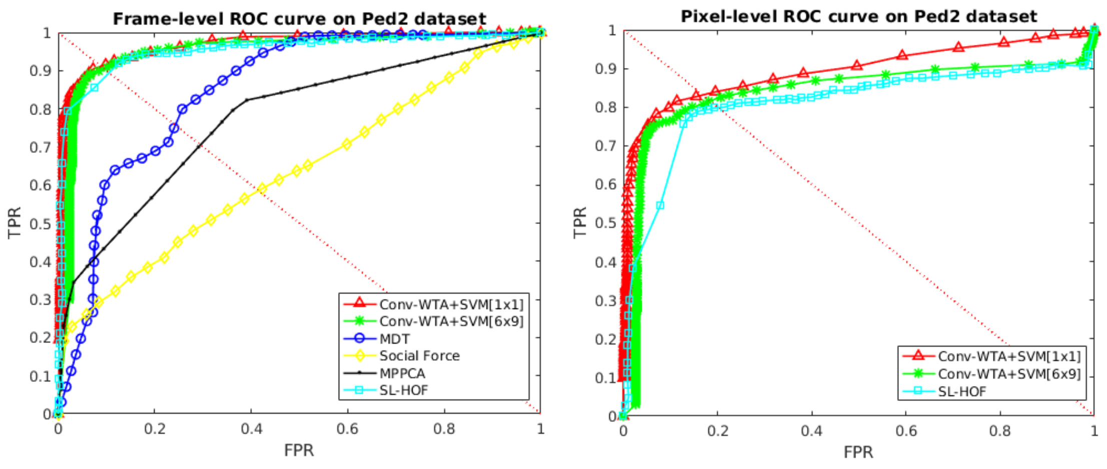


Figure 3.13: Frame-level and pixel-level evaluation on the UCSDPed2.

Our method achieves a significantly better EER and AUC results in both frame-level and pixel-level on UCSDPed2 with OCSVM$[1 \times 1]$ (Table 3.1). The pixel-level AUC is improved by around 8% over the SL-HOF [33]. The result is also better on Avenue with OCSVM$[8 \times 12]$ (Table 3.2), where the frame-level is improved around 1% over the SCL [16]. Moreover, the method obtains comparable results with OCSVM$[6 \times 9]$ on UCSDPed1 (Table 3.1).

Method	**UCSDPed1**		**UCSDPed2**	
	Frame level %	Pixel level %	Frame level %	Pixel level %
	EER/AUC	EER/AUC	EER/AUC	EER/AUC
Sparse Coding [2]	19/86	54/46.1	-	-
MDT [17]	25/81.8	55/44	25/85	55/-
MPPCA [18]*	40/59	82/-	30/77	-
Social Force Model [4]*	31/67.5	79/-	42/63	-
SCL [16]	15/91.8	40.9/63.8	-	-
SL-HOF [33]	18/87.5	**35**/64.4	9/95.1	19/81
AMDN [34]	16/**92.1**	40.1/67.2	17/90.8	-
Conv-AE [12]	27.9/81	-	21.7/90	-
Conv-WTA-OCSVM$[1 \times 1]$	27.9/81.3	46.8/56	**8.9**/**96.6**	**16.9**/**89.3**
Conv-WTA-OCSVM$[6 \times 9]$	**14.8**/91.6	35.8/66.1	9.5/95	18.4/83.9
Conv-WTA-OCSVM$[12 \times 18]$	15.9/91.9	35.7/**68.7**	11.2/92.8	21.2/80.9

Table 3.1: Performance comparison on UCSDPed1 and UCSDPed2. (* the results in [17] include replicated results for MPPCA [18] and Social Force Model [4] methods.)

As can be seen from Table 3.1, a finer sub-division gives better results on UCSDPed1, whereas the best results are obtained for no sub-division on UCSDPed2 (i.e. OCSVM$[1 \times 1]$). This may be explained by the greater variation in scale in UCSDPed1 than in UCSDPed2, leading to substantial variations in the patterns of motion as an object moves in depth through the scene. It may also be due to 'contextual' anomalies such as a pedestrian walking over grass that occupies only a portion of the scene. Finally, it is worth noting that a finer sub-division results in less training data for each one-class SVM, which may result in unexpected results where there is inadequate training data.

We also do evaluation using the VOC Pascal style objection detection evaluation method for CUHK Avenue dataset. A correct detection should satisfy the criterion of Intersection over Union IoU $> \upsilon$. One frame is true positive when IoU between its detection result and its ground-truth larger than predefined threshold υ. Table 3.3 shows that our results are better than SCL method [16] when increasing υ, our detections overlap well with the ground-truth.

Method	Frame level (%)		Pixel level (%)	
	EER	AUC	EER	AUC
SCL [16]*	-	80.9	-	-
Discriminative framework [19]	-	78.3	-	-
Conv-AE [12]	25.1	70.2	-	-
Conv-WTA + OCSVM[1×1]	28.2	78.1	50	50.7
Conv-WTA + OCSVM[4×6]	26.5	81	45.7	54.2
Conv-WTA + OCSVM[8×12]	**24.2**	**82.1**	**45.2**	**55**

Table 3.2: Performance comparison on the Avenue dataset. (* the results from [19] replicated SCL method [16])

υ	0.2	0.3	0.4	0.5	0.6	0.7	0.8
SCL [16]	70.0%	67.3%	63.3%	59.3%	57.5%	55.7%	54.4%
Ours	**70.5%**	**69.8%**	**67.8%**	**66.1%**	**64.3%**	**63.3%**	**63.1%**

Table 3.3: Detection accuracy (%) with IoU threshold υ on CUHK Avenue dataset. OCSVM[8×12] is used.

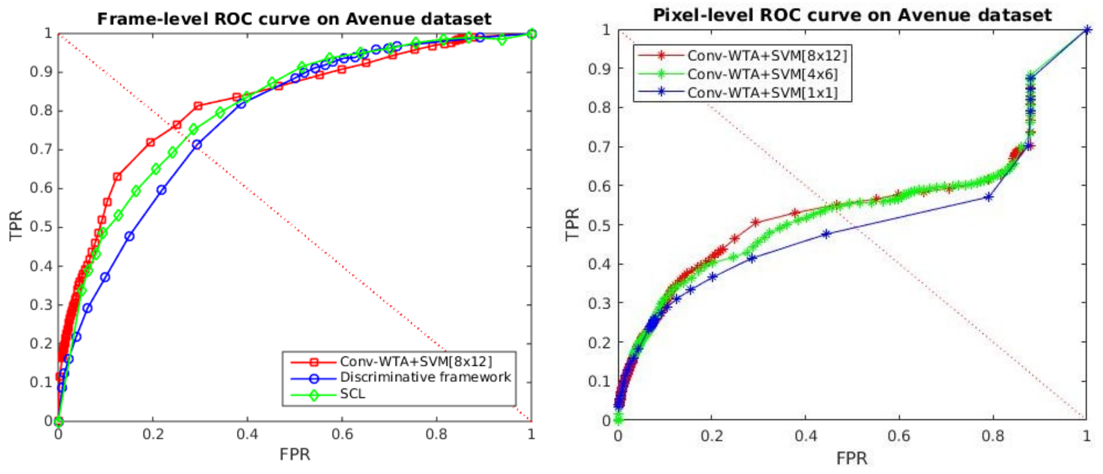


Figure 3.14: Frame-level comparison on the Avenue dataset.

We evaluate our convolutional WTA feature extractor with different patch sizes, different pooling sizes and time steps. The results are shown in next sections. Moreover, to evaluate the efficiency of the WTA step for learning feature representations in the autoencoder, we evaluate the model trained with and without the WTA spatial sparsity constraint.

3.3.3.2 Varying patch size.

In this section, we compare the EER/AUC with different patch sizes on UCSDPed1 and UCSDPed2 datasets. Table 3.4 shows our results with three different patch sizes and strides. Stride is the number of pixels by which the window is slid to extract patches. We chose the sizes around the common sizes use in the state of the art methods. Then a suitable value is chosen for a stride to make sure the image field of view is scanned. As can be seen in the Table, patch size of 24 gives the best results on both UCSDPed1 and UCSDPed2. In all these experiments, we use a global one-class SVM (i.e, OCSVM[1×1]) for UCSDPed2 and a local one-class SVM (OCSVM[6×9]) for UCSDPed1.

Patch size (stride)	**UCSDPed1**		**UCSDPed2**	
	Conv-WTA-OCSVM[6×9]		Conv-WTA-OCSVM[1×1]	
	Frame level %	Pixel level %	Frame level %	Pixel level %
	EER/AUC	EER/AUC	EER/AUC	EER/AUC
$16(7)$	$15.8/90.4$	$44.2/57.6$	$12.5/94.5$	$29.8/74.6$
$20(8)$	$\mathbf{14.7/92.3}$	$36.8/64.5$	$9.7/96.3$	$20.2/84.9$
$24(12)$	$15.4/91.4$	$\mathbf{33.9/66.9}$	$\mathbf{8.72/97}$	$\mathbf{16.6/89.8}$

Table 3.4: Performance comparison on UCSDPed1 and UCSDPed2 with different patch sizes.

3.3.3.3 Varying max-pooling size.

Max-pooling is used after training the autoencoder with WTA. Thus, we benefit from the sparse representation that WTA promotes and the dimensionality reduction of the coding for OCSVM. In this section, we evaluate the impact of using a max-pooling layer by varying the max-pooling size on UCSDPed1 (Table 3.5). We use patch size of 24×24 in the experiments.

Table 3.5 shows a comparison on UCSDPed1. The results are better with max-pooling size $p = 6$. Each point in the pooling output tensor can be considered as feature representations for a corresponding receptive field. Changing max-pooling size changes the receptive field's size. When the pooling size is reduced, the size of an encoding representation is increased. It can be seen that the feature representation of a patch size of 24×24 is obtained by concatenating feature representations of some overlapped smaller patches. Thus the feature representation considers spatial relationship between smaller neighbouring patches. However, reducing max-pooling size also increases the dimension of feature representation, affecting the efficiency of learning and testing with OCSVM. We use a max-pooling size $p = 6$ for comparing our results with the state of the art on the UCSD dataset (Table 3.1) and $p = 12$ for evaluating our frame-work on the Avenue dataset.

Max-pooling size p	Encoding representation	Subdivision	Frame level (%)		Pixel level (%)	
			EER	AUC	EER	AUC
$p = 12$	$1 \times 1 \times 128$	$[1 \times 1]$	28.4	81.1	47.1	52.8
		$[6 \times 9]$	15.5	91.3	**34.4**	65.7
		$[12 \times 18]$	16.2	91.5	35.5	67.7
$p = 6$	$2 \times 2 \times 128$	$[1 \times 1]$	27.9	81.3	46.8	56
		$[6 \times 9]$	**14.8**	91.6	35.8	66.1
		$[12 \times 18]$	15.9	**91.9**	35.7	**68.7**
$p = 4$	$3 \times 3 \times 128$	$[1 \times 1]$	27.9	80.8	46.8	55.4
		$[6 \times 9]$	15.3	91.1	38.6	63.2
		$[12 \times 18]$	16.7	91.4	38.1	65.9

Table 3.5: Performance comparison on UCSDPed1 with different kernel sizes and strides of max-pooling and different subdivisions.

3.3.3.4 Smoothness over number of frames

We do temporal averaging over number of frames to create more robust feature representations. In this section, we show the effectiveness of the smoothness over the final detection results. Results are obtain by using convolutional WTA feature extractor on patches size of 24×24, max-pooling size $p = 12$ with OCSVM$[6 \times 9]$ for UCSDPed1 and with OCSVM$[1 \times 1]$ for UCSDPed2. As can be seen in Table 3.6, performance is improved when increasing the number of frames.

	UCSDPed1		**UCSDPed2**	
	Conv-WTA-OCSVM[6×9]		Conv-WTA-OCSVM[1×1]	
Number of frames	Frame level % EER/AUC	Pixel level % EER/AUC	Frame level % EER/AUC	Pixel level % EER/AUC
1 frame	24.5/83.5	53/47.4	12.6/93.6	27.2/77.9
3 frame	18.1/89.2	43.3/58.8	10/96.2	19.4/87
5 frame	15.5/91.3	34.4/65.7	8.7/97	16.6/89.8
7 frame	**15.2/91.6**	**33.1/67.1**	6.3/98	14.3/91.6
9 frame	15.8/91.4	33.6/67	**5.3/98**	**12.1/92.9**

Table 3.6: EER/AUC for different temporal smoothing windows.

3.3.3.5 Efficiency of the Winner-Take-All sparsity constraint

Our method consists of many steps such as foreground extraction, feature extraction and OCSVM with different subdivisions. In order to evaluate the efficiency of the WTA sparsity constraint for feature representation learning, we do experiments in which we keep foreground extraction and OCSVM steps unchanged and change the feature extractor. Table 3.7 compares the results obtained with the convolutional feature extractor and the convolutional WTA feature extractor on UCSDPed1 and UCSDPed2. In experiments, we use patch size of 24×24, temporal averaging over 5 frames, max-pooling size $p = 12$, OCSVM[1×1] for UCSDPed2 and OCSVM[6×9] for UCSDPed1. As shown in Table 3.7, the convolutional WTA feature extractor outperforms the convolutional feature extractor, where pixel-level EER and AUC are significantly improved.

	UCSDPed1		**UCSDPed2**	
	Conv-WTA-OCSVM[6×9]		Conv-WTA-OCSVM[1×1]	
Method	Frame level % EER/AUC	Pixel level % EER/AUC	Frame level % EER/AUC	Pixel level % EER/AUC
Conv feature extractor	23.2/84.2	49.4/48.6	14.1/92.7	33.7/69.5
Conv-WTA feature extractor	**15.4/91.4**	**33.9/66.9**	**8.7/97**	**16.6/89.8**

Table 3.7: Impact of the WTA constraint.

3.3.4 Qualitative Analysis

In this section, we present a qualitative analysis of the results provided by using the convolutional WTA feature extractor and OCSVM with subdivisions on UCSDPed1, UCSDPed2 and CUHK Avenue datasets. We use a patch size of 24×24, OCSVM[6×9] for UCSDPed1, OCSVM[1×1] for UCSDPed2 and OCSVM[8×12] for Avenue.

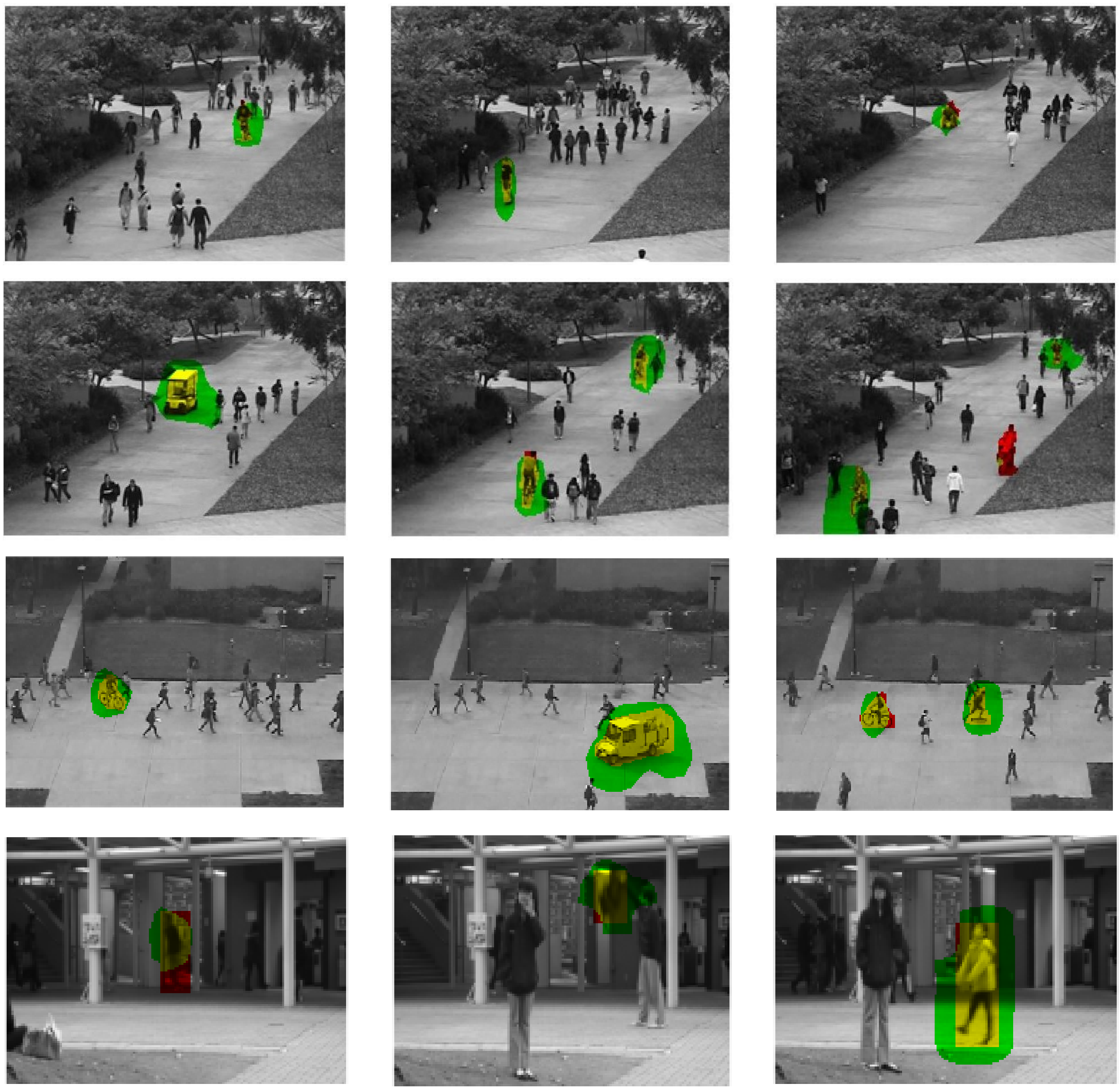
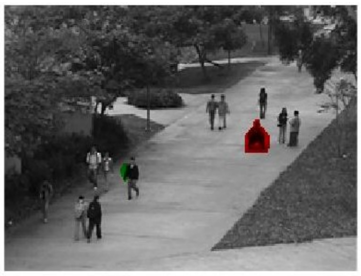
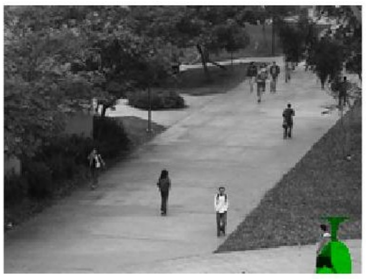


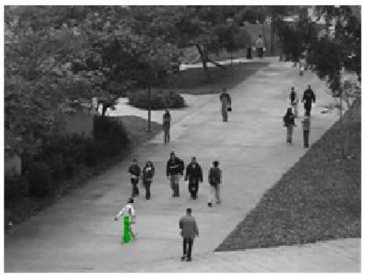
Figure 3.15: Detection results on the UCSDPed1 (first 2 rows), the UCSDPed2 (third row) and the CUHK Avenue dataset (fourth row).

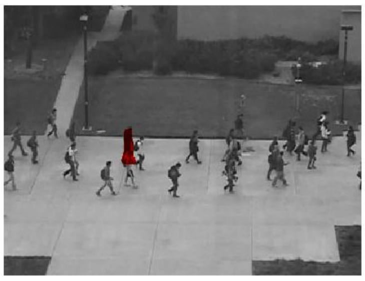
Figure 3.15 shows the detection results on three datasets. Our method can detect the anomalies of a biker, skater, car and wheelchair on UCSD dataset. Moreover, running, loitering, throwing object and jumping on CUHK Avenue dataset are also detected. In these figures, pixels that have been correctly predicted as anomalous are shown in yellow; anomalous pixels that have been missed are shown in red, and pixels that have been incorrectly predicted as anomalous are shown in green.

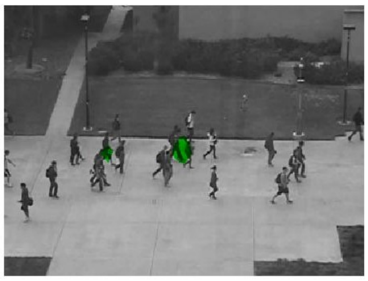
We also observe missed detections and false positives. Figure 3.16 (a), (d) and (g) show some missed detections in three datasets in which (a) the wheelchair has small anomaly score when it goes further; (b) a skater with slow motion sometimes looks like a normal pedestrian and a stand-still person in (g) makes no optical flow. It reveals the disadvantage of using optical flow, detection relies on motion.

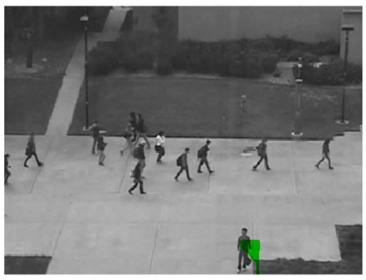

(a)

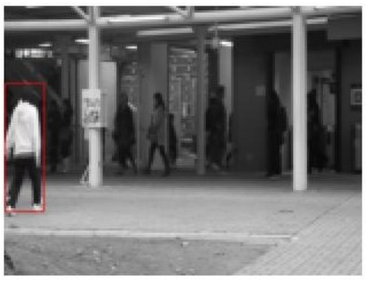

(b)

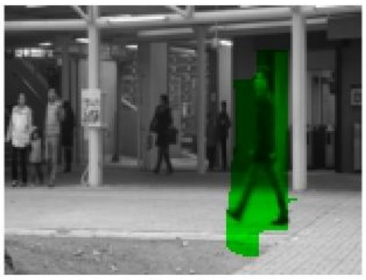

(c)


(d)


(e)


(f)


(g)


(h)

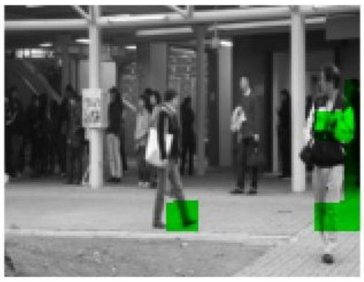

(i)

Figure 3.16: False detection results on the UCSDPed1 (first row), the UCSDPed2 (second row) and the Avenue dataset (third row).

Figure 3.16 (b), (c), (e), (f), (h) and (i) show examples of false positives. In UCSDPed1, horizontal movement rarely appears in the training set. Therefore, this motion in the test set is detected as anomalies as shown in Figure 3.16(b) and (c). In UCSDPed2, groups of pedestrians are detected as anomalies as illustrated in Figure 3.16(e). Moreover, a motion, which rarely happens in the training set, is also detected as an anomaly as in Figure 3.16 (f). Similar to UCSDPed1, horizontal movements in the area close to the camera in CUHK Avenue are detected as anomalies since they rarely appear in the training set (Figure 3.16 (h) and (i)).

3.3.5 Number of parameters and running time

In this section, we compare the number of parameters of our convolutional WTA feature extractor with the feature extractor using fully connected autoencoder from Xu et al. [34] in Table 3.8. We only show the number of parameters of the encoders that are used for feature extraction. As illustrated in Table 3.8, convolutional layers reduce the number of parameters significantly.

Layer	Conv-WTA Feature Extractor	Fully-connected Feature Extractor [34]
1	$5 \times 5 \times 2 \times 128$	$15 \times 15 \times 2 \times 2048$
2	$5 \times 5 \times 128 \times 128$	2048×1024
3	$5 \times 5 \times 128 \times 128$	1024×512
4		512×256
# parameters	$825,600$	$3,674,112$

Table 3.8: The details of the number of parameters for each autoencoder.

	Optical Flow	Feature Extraction	OCSVM	Total
Without GPU	0.6302	3.7996	0.0055	4.4353
With GPU	0.6302	**0.0857**	0.0055	**0.7214**

Table 3.9: Testing time (second/frame) without and with GPU.

We also present running times of our method on a computer (CPU - Intel Core i7-4790) with and without the use of GPU (GeForce GTX Titan X). We run our experiments using Matlab2016a, the toolbox from Liu [109] for computing optical flow, Matconvnet [110] for deep learning and libSVM [111] for one-class SVM. GPU, which is more specialized at performing matrix operation, speeds up the feature extraction process significantly as illustrated in Table 3.9. However, the testing time is still affected by optical flow computation.

3.4 Conclusions

In this chapter, we present a framework that uses a convolutional Winner-Take-All autoencoder to learn motion feature representations for anomaly detection. The convolutional WTA feature extractor extracts compressed, robust motion feature representations. Moreover, the combination of this motion feature representation with a local application of one-class SVM gives competitive performance on two challenging datasets in comparison to existing state-of-the-art methods.

However, learning motion feature representations on optical flow requires to compute optical flow as a preprocessing step, which slows down the running time. In addition, we use one-class SVM for training a model of normality and detecting anomalies. The system cannot be trained end-to-end. Therefore, in the next chapter, we present a method that can be trained end-to-end to learn normal temporal dynamics on video frames.

Chapter 4

Convolutional Long Short-Term Memory for anomaly detection

4.1 Introduction

In this chapter, we present our proposed method for video anomaly detection based on sequence-to-sequence prediction and reconstruction, using a layered convolutional Long Short-Term Memory (convLSTM) encoder-decoder network. As in previous work [12], anomalies arise as spatially localised failures in reconstruction. Detecting anomalies using reconstruction error, the encoder-decoder network can be trained end-to-end without the combination of a one class SVM on top of pre-learned feature representations.

In our work, we adopt a convLSTM encoder-decoder to learn normal temporal dynamics from sequences of successive frames. Convolutional LSTMs have already been employed for anomaly detection. However, the temporal unit in the approach by Chong et al. [13] is applied on the final spatial stage, which encodes high level representations. Interleaving RNNs between spatial convolution layers has recently been shown to improve performance on precipitation now-casting [112]. The model can learn temporal information on hierarchical spatial representations from low-level to high-level. We adopt the same architecture, except that we remain with convolutional LSTMs [104] instead of the newly proposed trajectory Gated Recurrent Unit [112]. A sequence-to-sequence convolutional LSTM encoder-decoder can be trained for both reconstruction and prediction, which can

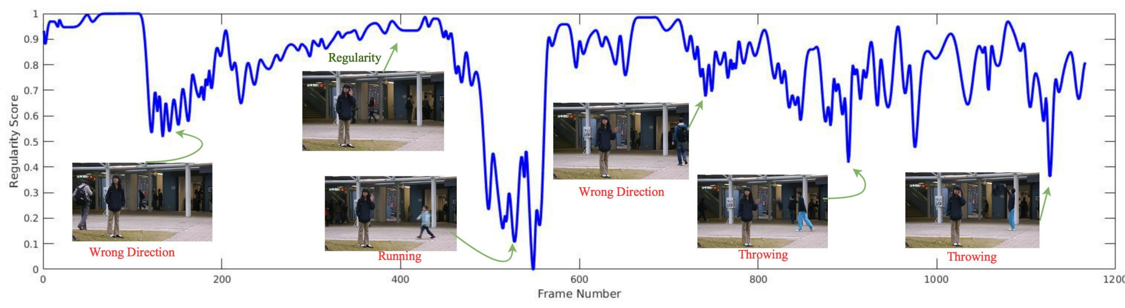


Figure 4.1: Regularity scores obtained from an extract from the CUHK Avenue dataset[16]. The regularity score drops when an abnormal event appears.

then be used for anomaly detection.

In experiments with five benchmark datasets, we show that using prediction gives superior performance than using reconstruction. Moreover, the use of convLSTM layers helps to improve performance over the similar architectures without these layers. We also compare performance with different length of input/output sequences. Overall, our results using prediction are comparable with the state of the art on the benchmark datasets.

4.2 Architecture

For ease of explanation, we focus on using the model for prediction. At each time step, the network takes as input a sequences of τ video frames $\mathcal{F}_{t-\tau+1}, ..., \mathcal{F}_t$, and generates an output of the same size, representing a τ-step prediction into the future $\mathcal{F}_{t+1}, ..., \mathcal{F}_{t+\tau}$. The model for reconstruction is obtained by using input sequences in reverse order as target sequences in the same architecture. Figure 4.2 illustrates the convLSTM encoder-decoder structure for future prediction, with $\tau = 5$.

In order to evaluate the impact of convLSTM layers on temporal dynamics learning, we also introduce the a simpler convolutional encoder-decoder as shown in Figure 4.3, in which we replace convLSTM layers by convolutional layers with skip-connections [39, 113]. Skip connections help to pass the image details to the top layers in the decoder.

The convLSTM encoder-decoder and the convolutional encoder-decoder structures consist of two networks, an encoding network and a decoding network. In the convLSTM encoder-decoder, the encoder contains convolutional layers and the decoder consists of deconvolutional layers. Leaky ReLU with negative slope equal to 0.2 [86] is used after each convolutional and deconvolutional layer (except in the last deconvolutional layer). To learn temporal dynamics with convolutional LSTM, an encoding RNN is interleaved between convolutional layers and a decoding RNN is interleaved between deconvolutional

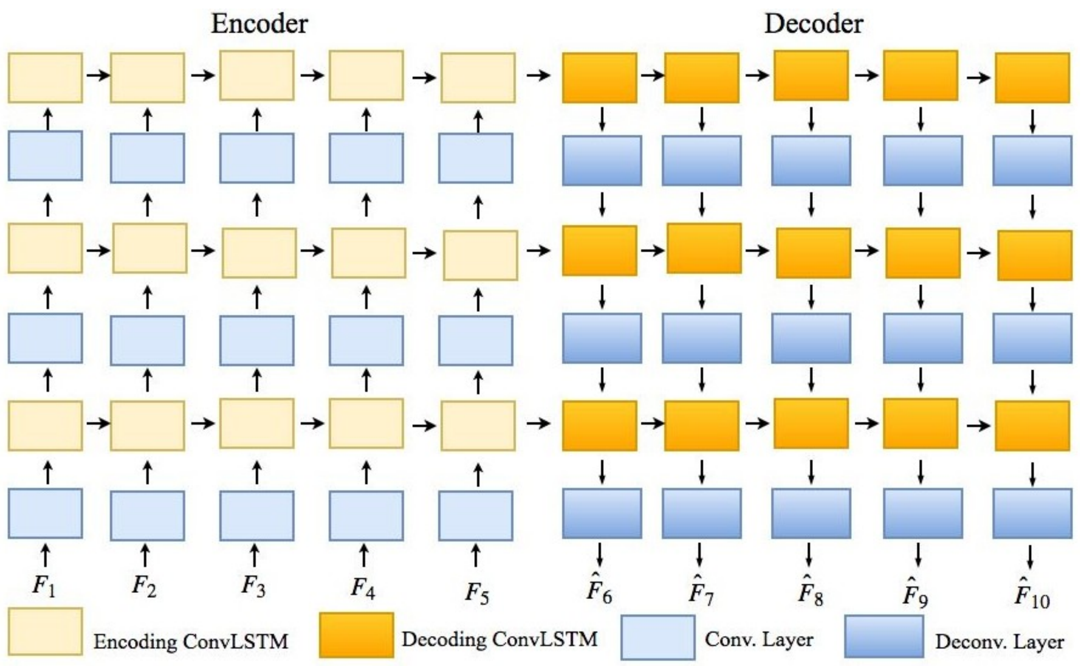


Figure 4.2: The convLSTM encoding-decoding structure used for future prediction with $\tau = 5$.

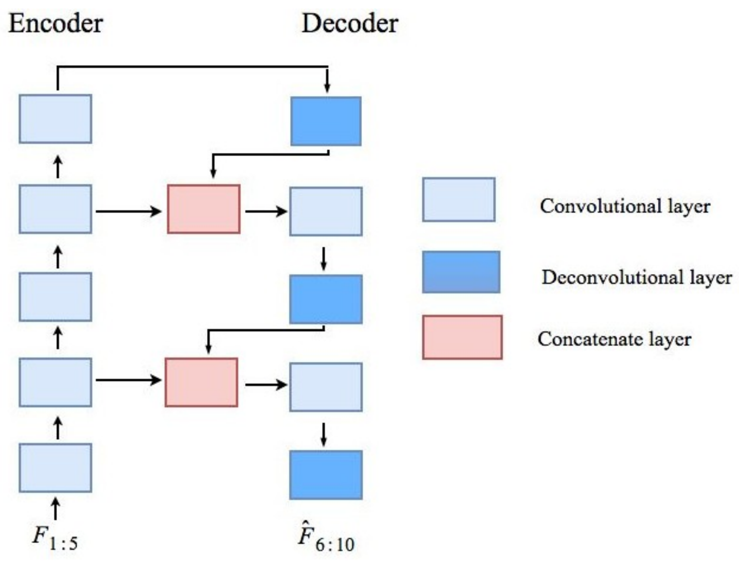


Figure 4.3: The convolutional encoding-decoding structure with skip connections used for future prediction with $\tau = 5$.

layers. The details of the convLSTM encoder-decoder architecture are shown in Table 4.1 and Table 4.2.

In the convolutional encoder-decoder, to pass image details from an encoder to a decoder, we use 'concatenate layers' to concatenate output tensors of deconvolutional layers with output tensors of their corresponding convolutional layers in the encoder. The detail of the convolutional encoder-decoder architecture is illustrated in Table 4.3.

Inputs for the convLSTM encoder-decoder, the convolutional encoder-decoder and the convolutional encoder-decoder are presented in the next section.

4.2.1 Input data layer

In anomaly detection, the input data is a video clip consisting of multiple frames. The input to our model is a video volume which consists of τ frames. Before stacking frames together to form input video volumes, each frame is extracted from the raw video, converted to a gray-scale image by weighting the sum of the R, G and B components. We use the same spatial input size as Hasan et al. [12], therefore we resize gray-scale images to 227×227 using bilinear interpolation. Then, the pixel values are scaled to the range $[0, 1]$.

As used previously in the work of Hasan et al. [12], we stack T frames in the channel dimension to construct the input to the convolutional encoder-decoder network. The convolutional layer treats the temporal dimension as the channel information of the input. Since we use a gray-scale image, our video sequence has size of $227 \times 227 \times \tau$. For the convLSTM encoder-decoder, we construct the input as a video volume by stacking τ frames in the 4^{th} dimension (also called temporal dimension). Thus our video volume has size of $227 \times 227 \times 1 \times \tau$.

As the number of parameters of the network is large, a large amount of data is required to train the model. However, the number of training data in anomaly datasets is not large enough. Therefore, data augmentation methods are employed to avoid over-fitting. In our work, we implement two methods for data augmentation. Firstly, we follow the same approach as Hasan et al. [12] in which more video volumes are created by concatenating successive frames with skipping strides. Three values of skipping stride (stride-1, stride-2, stride-3) are used to construct τ-sized volumes. With stride-1, τ frames are consecutive, while with stride-2 and stride-3, we skip one and two frames, respectively. We name this method as aug1.

However, increasing the temporal stride increases the speed of moving objects in the input volume. Since speed is an important feature in many anomaly detection scenarios, it may affect a detection result. Therefore, we adopt a second data augmentation method (aug2) that uses standard image processing techniques such as Gaussian blurring, image sharpening and histogram equalization. Moreover, we randomly crop video frames with a spatial size of 100×100 and then resize these cropped windows to form input volumes. Augmentation methods are only employed on training data.

The input data then are fed to networks to train our models. In the next section, we shows more detail about the encoder-decoder structures.

4.2.2 Convolutional and Deconvolutional layer

In both convLSTM encoder-decoder and convolutional encoder-decoder structures, convolutional layers act as feature extractors which preserve spatial relationship between pixels within video frames. In our architectures, we use convolutional layers with stride greater than 1 to abstract the activations, which allows the network to learn its own spatial downsampling. Stacking multiple convolutional layers in the encoding side coarsens output maps, reducing spatial size from the input by a factor equal to the stride. By choosing the number of convolutional filters, the output maps can be compressed or over-completed. Unlike the early work of Hasan et al. [12], the output activations in our networks are compressed after each convolutional layer.

The decoder, which consists of deconvolutional layers, connects the compressed coarse output back to dense pixels. Since deconvolution reverses the forward and backward passes of convolution, up-sampling is performed. The learned filters in the deconvolutional layers correspond to bases to predict or reconstruct shape of an input object. Similar to convolutional layers, a hierarchical structure of deconvolutional layers are used to capture different level of shape details. We use a leaky ReLU layer after each convolutional and deconvolutional layer, except the last deconvolutional since it limits the range of the output.

The details about the number of layers, the kernel size, the spatial size of input/output tensors, the number of input/output channels, padding and stride for each layer in the convolutional and convolutional LSTM encoder-decoders are described in Table 4.1, 4.2 and 4.3. In these tables, "Kernel" is the size of convolutional and deconvolutional filters, "In Res" and "Out Res" is the input and output spatial resolution, respectively. The depth of input and output tensors is shown under "Ch I/O". In the convolutional layer, "Pad" is the number of zeros which are padded to the boundaries of an input tensor and "Stride" is the number of pixels by which filters shift. In the deconvolutional layer, "Pad" is the number of pixels in the boundaries of an output tensor which are cropped and "Stride" is an up-sampling factor.

Type	Kernel	Stride	Pad	Ch I/O	In Res	Out Res
Conv-ReLU	11x11	4	0	1/16	227x227	55x55
ConvLSTM	3x3	1	1	16/16	55x55	55x55
Conv-ReLU	3x3	2	0	16/32	55x55	27x27
ConvLSTM	3x3	1	1	32/32	27x27	27x27
Conv-ReLU	3x3	2	0	32/64	27x27	13x13
ConvLSTM	3x3	1	1	64/64	13x13	13x13
ConvLSTM	3x3	1	1	64/64	13x13	13x13
Deconv-ReLU	3x3	2	0	64/32	13x13	27x27
ConvLSTM	3x3	1	1	32/32	27x27	27x27
Deconv-ReLU	3x3	2	0	32/16	27x27	55x55
ConvLSTM	3x3	1	1	16/16	55x55	55x55
Deconv	11x11	4	0	16/1	55x55	227x227

Table 4.1: The details of the convolutional LSTM encoder-decoder model with 12 layers. The two dimensions in "Kernel", "In Res" and "Out Res" represent for height and width.

Type	Kernel	Stride	Pad	Ch I/O	In Res	Out Res
Conv-ReLU	11x11	4	0	1/16	227x227	55x55
ConvLSTM	5x5	1	2	16/16	55x55	55x55
Conv-ReLU	3x3	2	0	16/32	55x55	27x27
ConvLSTM	3x3	1	1	32/32	27x27	27x27
ConvLSTM	3x3	1	1	32/32	27x27	27x27
Deconv-ReLU	3x3	2	0	32/16	27x27	55x55
ConvLSTM	5x5	1	2	16/16	55x55	55x55
Deconv	11x11	4	0	16/1	55x55	227x227

Table 4.2: The details of the convolutional LSTM encoder-decoder model with 8 layers.

Type	Kernel	Stride	Pad	Ch I/O	In Res	Out Res
Conv-ReLU	11x11	4	0	5/16	227x227	55x55
Conv-ReLU	3x3	1	1	16/16	55x55	55x55
Conv-ReLU	3x3	2	0	16/32	55x55	27x27
Conv-ReLU	3x3	1	1	32/32	27x27	27x27
Conv-ReLU	3x3	2	0	32/64	27x27	13x13
Deconv-ReLU	3x3	2	0	64/32	13x13	27x27
Concat				-/64	27x27	27x27
Conv-ReLU	3x3	1	1	64/32	27x27	27x27
Deconv-ReLU	3x3	2	0	32/16	27x27	55x55
Concat				-/32	55x55	55x55
Conv-ReLU	3x3	1	1	32/16	55x55	55x55
Deconv	11x11	4	0	16/5	55x55	227x227

Table 4.3: The details of the 2D convolutional encoder-decoder with skip connections.

In the convLSTM encoder-decoder, the convolutional and deconvolutional layers treat τ frames in the input volumes as τ separate frames. For example, after the first convolutional layer, we obtain an output tensor of size $55 \times 55 \times 16 \times \tau$. The temporal information is captured using a RNN with a convolutional LSTM. We discuss the convolutional LSTM in the next section.

4.2.3 Recurrent Neural Network using Long Short-term Memory

RNNs are powerful learning models that are often used for learning sequential tasks. Since input volumes contain dynamic content, the variation between frames may encode additional information that is useful for anomaly detection.

Given an input $\mathbf{x}_t$ in the sequence, the standard RNN cell computes the hidden vector $\mathbf{h}_t$ and output $\mathbf{y}_t$ by iterating the following equations for τ time steps:

$$\begin{aligned} \mathbf{h}_t &= g(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t &= g(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o) \end{aligned} \tag{4.1}$$

where g is the activation function, $\mathbf{W_{xh}}$, $\mathbf{W_{hh}}$, $\mathbf{W_{ho}}$ denote the input-hidden, the hidden-output and the hidden-output weight matrices, respectively. $\mathbf{b}_h$ and $\mathbf{b}_o$ denote the hidden and the output bias vectors, respectively. It can be difficult to train RNN models to learn long-term dynamics. This limitation is partly because of the vanishing and the exploding gradients [98] which result from propagating the gradients down through many layers, each layer corresponds to a time step.

4.2.3.1 Long Short Term Memory

The Long Short-Term Memory (LSTM) architecture [98] introduces memory cells to store and output information, allowing it to better exploit long range dependencies in the data. Figure 4.4 illustrates a diagram of the LSTM memory cell from [99], which is derived from the original LSTM unit [98].

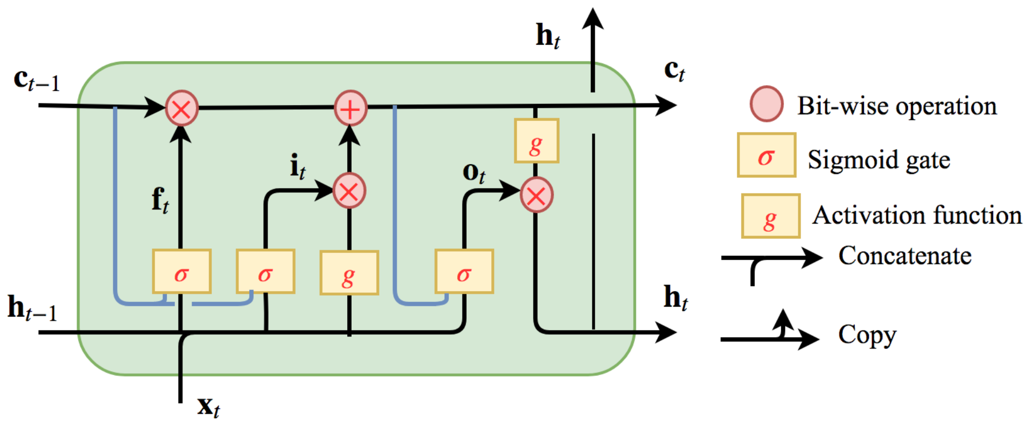


Figure 4.4: A diagram of an Long Short-term Memory cell, an activation function can be tanh or ReLU.

Each LSTM unit has a cell which has a state $\mathbf{c}_t$ at time t. The memory unit is accessed for reading or modifying by three sigmoid gates - input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$ and output gate $\mathbf{o}_t$. At each time step, it receives inputs from two sources: the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. Additionally, each gate also considers the cell state $\mathbf{c}_{t-1}$ of its cell block, as its third input. Peephole connections, which are the links between a cell and gates, allow the gates to access the memory cell $\mathbf{c}$ at time $t-1$. Peephole connections are illustrated as blue lines in Figure 4.4. Every time a new input comes, its information will be accumulated to the cell $\mathbf{c}_t$ if the input gate $\mathbf{i}_t$ is activated. The sigmoid layer outputs numbers lie within the range $[0, 1]$, describing how much of each component can get through. Also, the forget gate $\mathbf{f}_t$ defines how much of the previous state $\mathbf{c}_{t-1}$ can be kept in the new state $\mathbf{c}_t$. Finally, the final state (the output of LSTM cell) $\mathbf{h}_t$ is based on the cell state but with the control of the output gate $\mathbf{o}_t$. These updates are summarized as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ g(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \circ g(\mathbf{c}_t)
\end{aligned} \tag{4.2}$$

where '$\circ$' denotes the Hadamard product, σ is the logistic sigmoid function, all activation vectors $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t, \mathbf{c}_t$ have the same size as the hidden vector $\mathbf{h}_t$. The weight matrix subscripts have the obvious meaning, for example $\mathbf{W}_{hi}$ is the hidden-input gate matrix, $\mathbf{W}_{xo}$ is the input-output gate matrix. The weight matrices from the cell to gate vectors (e.g. $\mathbf{W}_{c*}$) are diagonal. The activation function g can be tanh or ReLU.

4.2.3.2 Convolutional Long Short Term Memory

Fully connected LSTM units unfold the inputs to 1D vectors. Thus it loses the spatial information and is difficult to apply on high dimensional input tensors. Convolutional LSTM cells [104] are used to learn temporal dynamics in our convolutional LSTM encoder-decoder. Since convolutional LSTM replaces matrix products in Equation 4.2 with convolutions, it also learns spatial information with fewer weights. The updates in the convolutional LSTM cell at time t are given as following:

$$\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1} + \mathbf{W}_{ci} \circ \mathbf{C}_{t-1} + \mathbf{b}_i) \\
\mathbf{F}_t &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1} + \mathbf{W}_{cf} \circ \mathbf{C}_{t-1} + \mathbf{b}_f) \\
\mathbf{C}_t &= \mathbf{F}_t \circ \mathbf{C}_{t-1} + \mathbf{I}_t \circ g(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}_{t-1} + \mathbf{b}_c) \\
\mathbf{O}_t &= \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1} + \mathbf{W}_{co} \circ \mathbf{C}_t + \mathbf{b}_o) \\
\mathbf{H}_t &= \mathbf{O}_t \circ g(\mathbf{C}_t)
\end{aligned} \tag{4.3}$$

where $*$ represents the convolution operation, and $\circ$ the Hadamard product. g is the activation function such as tanh or ReLU. $\mathbf{I}_t, \mathbf{F}_t, \mathbf{C}_t, \mathbf{O}_t, \mathbf{H}_t \in \mathbb{R}^{H\times W\times C_o}$ are the input gate, forget gate, cell state, output gate and hidden state, respectively. $\mathbf{X} \in \mathbb{R}^{H\times W\times C_i}$ is the input tensor. H and W are the height and width of the state and input tensor; C_o and C_i are the channel size of the state and input tensor, respectively. We choose tanh for the activation function g.

Interleaving multiple convLSTM layers between convolutional layers (as illustrated in

Figure 4.2) helps our convLSTM encoder-decoder learn spatio-temporal dynamic information at different levels. The high level states capture global spatial-temporal representations while the lower level states retain the detail of local spatio-temporal representations. After the last frame in the input sequence is read, the decoding convLSTMs take the last encoding states and output an estimate of the target volume. The encoding states at two lower levels are also used as initial states for the two corresponding decoding convLSTMs, which pass feature maps containing image details. These connections share the same aim as skip connections [39, 113]. Convolutional layers with strides greater than 1 in the encoder may lose image details, making it difficult to estimate the output with the deconvolutional layers. These initializations help the decoding convLSTMs and deconvolutional layers to output a better estimate.

4.3 Optimization and Initialization

The optimization objective of the model is defined by the prediction error, with an augmented L_2 regularization term. The weights and biases of layers in the model are learned by minimizing the regularized least mean squared error as follows:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2N\tau}\sum_{n=1}^{N} \|\boldsymbol{\theta}_n - \hat{\boldsymbol{\theta}}_n\|_2^2 + \frac{\lambda}{2}\sum_{l} \|\mathbf{W}\|_2^2 \tag{4.4}$$

where $\hat{\boldsymbol{\theta}}_n$ is the predicted volume from the model, $\boldsymbol{\theta}_n$ is the target volume of τ frames, N is the size of a mini batch and $\mathbf{W}$ is the parameters of the model. The first term is the squares prediction error and the second term is to regularize the weights. λ is a hyper-parameter used to balance the importance of the two terms.

The weights in each convolutional layer are initialized using the Xavier algorithm [114] which automatically determines the scale of initialization based on the number of input and output neurons. This initialization makes sure the weights are not too small or not too large, keeping the signal in a reasonable range of values through many layers. We train all models with the default setting for the Xavier algorithm in the Caffe toolbox [116]; weights are initialized by sampling from the uniform distribution $U[-\sqrt{\frac{3}{n}}, \sqrt{\frac{3}{n}}]$ in which $n = k^2c$, where k is being equal to the spatial filter size and c is the number of input channels. We initialize the weights for convLSTM layers using a zero-mean Gaussian distribution with a fixed standard deviation of 0.01. The biases for all layers are initialized to zero. The input-to-hidden and hidden-to-hidden convolutional filters in the convLSTM layers are the same size as illustrated in Table 4.1 and Table 4.2.

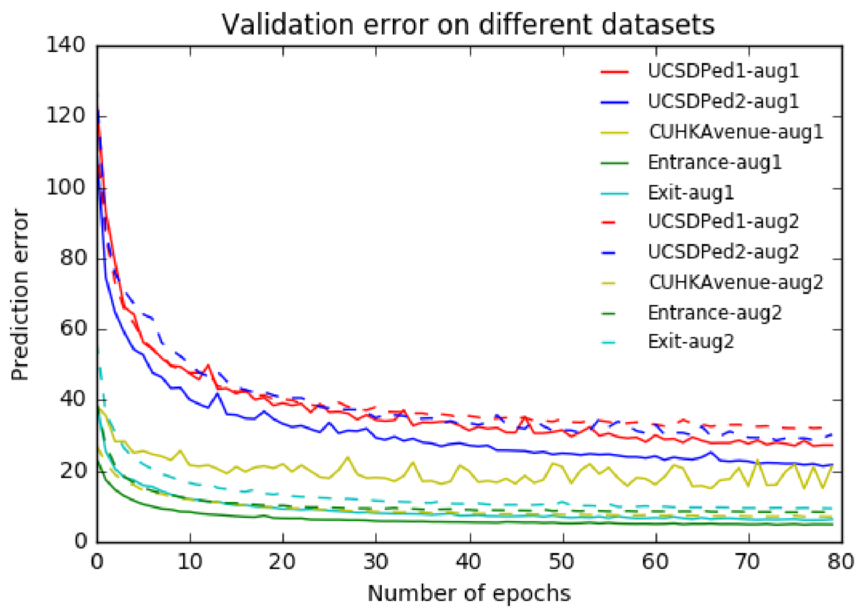


Figure 4.5: The validation error of the convLSTM encoder-decoder trained on each dataset.

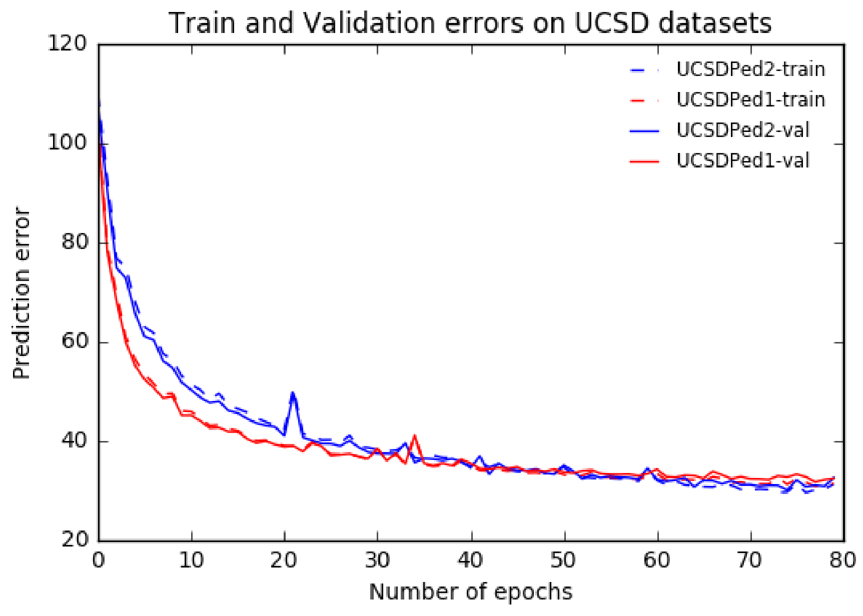


Figure 4.6: The train and validation errors of the convLSTM encoder-decoder trained on UCSDPed1 and UCSDPed2.

We use Adam [115] to optimize the cost function of Equation 4.4 with batch size $N = 4$, momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, weight decay $\lambda = 5 \times 10^{-4}$ [37].

We train our networks separately on each dataset so that the model can learn the specific normal patterns in each dataset. An event may be normal in one dataset but abnormal in another. For example, people going towards the turnstile to enter the subway station is normal in the Subway Entrance dataset but abnormal in the Subway Exit dataset. We start training the models with a learning rate of 10^{-4}. After 80 epochs, we stop training and use the models for anomaly detection. Figure 4.5 and 4.6 show train error on UCSD dataset and validation error on five datasets with two data augmentation methods in which the errors become smoother after around 80 epochs.

4.4 Regularity score for anomaly detection

Once the model is trained, we compute the prediction error between each estimated frame $\hat{\mathcal{F}}_i$ and the target frame $\mathcal{F}_i$ in the testing video volume, then sum up all τ frames to form the prediction error for this volume as follows:

$$e(t) = \sum_{i=t+1}^{i=t+\tau} ||\hat{\mathcal{F}}_i - \mathcal{F}_i||_2 \tag{4.5}$$

Then, we normalize the prediction error to compute the regularity score $r(t)$ of the testing volume [12]:

$$r(t) = 1 - \frac{e(t) - \min_{t'} e(t')}{\max_{t'} e(t')} \tag{4.6}$$

where $\min_{t'} e(t')$ and $\max_{t'} e(t')$ are calculated over the prediction errors of all volumes in the same test video. The test volume is abnormal if its regularity score $r(t)$ is less than a threshold.

We also use the same architecture for reconstruction in our experiments. Instead of using the next τ frames as the target volume, we use the input volume in reverse order as the target. Replacing the target volume $\mathcal{F}_i$ in Equation 4.5, we obtain the reconstruction error and use it for anomaly detection with the reconstruction model.

4.5 Experiments

We evaluate our method both quantitatively and qualitatively. We modify and use the Caffe toolbox [116] for our experiments, where models are trained on high performance computing clusters with NVIDIA Tesla P100 GPUs and tested on a desktop computer with CPU (Intel core i7-4790) and GPU (GeForce GTX TITAN X).

In this section, we compare the performance results on anomaly detection with different architectures and different setups as following:

- (1) We evaluate two architectures with different number of layers, one with 12 layers (Table 4.1) and one with 8 layers (Table 4.2). We also change the data augmentation method: aug1, aug2. In these experiments, we use the same value $\tau = 5$ and the decoding is for future prediction.
- (2) We do experiments with different decoders: reconstruction, future predictor. We use $\tau = 5$ and aug1 data augmentation method.
- (3) We do evaluation with different values of time-step τ: $\tau = 2, 5, 8$. We use aug1 for data augmentation.
- (4) We do experiments with the convolutional encoder-decoder with skip connections (the network's architecture in Table 4.3). We implement both decoders: reconstruction and future prediction. All experiments use aug2 for the data augmentation method.

4.5.1 Datasets

We train our models on three of the most commonly used datasets: UCSD (UCSDPed1 and UCSDPed2) [17], CUHK Avenue [16] and Subway (Entrance and Exit) [107]. UCSD and Avenue have separate training videos which contain mostly normal events and test videos, thus we use the training set for training models and then evaluate on test sets. With the Subway dataset, the first 15 minutes of Subway Entrance and the first 5 minutes of Subway Exit are used for training. In all datasets, the training data includes a few irregular events.

4.5.2 Anomalous event detection

Two performance metrics are employed for evaluation and comparison with state of the art results: Equal Error Rate (EER) and Area Under the ROC Curve (AUC). The regularity score of each volume determines whether it is normal or abnormal. We follow the intuition that video volumes containing normal events generate high regularity scores since they are similar to training data. A video sequence containing an anomaly gives a lower score. Setting different thresholds on the regularity score, we classify testing sequences into those that contain an anomaly and those that do not. These results are compared with ground-truth to give the equal error rate (EER) and area under the curve (AUC) of the resulting ROC curve (TPR versus FPR) generated by varying an acceptance threshold in a range of $[0, 1]$. Good performance has a low EER and high AUC. Two challenges in evaluating anomaly detection methods are (1) ensuring the evaluation datasets contain sufficient variety to build a representative model of normality, and (2) defining a ground-truth by subjective assessment, particularly in labelling marginal cases.

		UCSDPed1	UCSDPed2	CUHK Avenue
aug1	# training data	$18,000$	$6,000$	$41,000$
	# validation data	462	738	$4,072$
	# epochs	80	80	80
aug2	# training data	$30,000$	$11,000$	$55,000$
	# validation data	$2,470$	$1,030$	$5,736$
	# epochs	80	80	80

Table 4.4: Number of training data and training epochs correspond to each dataset.

Setup		EER/AUC (%)		
		UCSDPed1	UCSDPed2	CUHK-Avenue
Reconstruction	12 layers, aug1, $\tau = 5$	$28.9/75.6$	$17.1/87.5$	$26.1/81.4$
	12 layers, aug2, $\tau = 5$	$29.4/76$	$19.3/87$	$24.6/82.5$
Prediction	12 layers, aug1 $\tau = 2$	$27.1/78.3$	$21.1/86.1$	$22.5/85.1$
	12 layers, aug1 $\tau = 5$	$25.1/80.8$	$14.4/92.3$	$\mathbf{22.4/84.8}$
	12 layers, aug1 $\tau = 8$	$26.5/79$	$18.5/89.6$	$23.2/83.2$
	12 layers, aug2, $\tau = 5$	$\mathbf{24.4/81.8}$	$\mathbf{13.1/92.8}$	$22.7/84.6$
	8 layers, aug2, $\tau = 5$	$25.2/81.4$	$14.5/92.5$	$22.8/84.3$
	8 layers, aug1, $\tau = 5$	$27.7/77.6$	$14.5/92$	$23.4/83.5$

Table 4.5: Comparison of EER/AUC with different architectures and setups of the convLSTM encoder-decoder. τ is the number of frames in an input volume and a target volume.

Table 4.5 shows the results when different architectures and setups of the convolutional LSTM encoder-decoder are used. The model trained for future prediction gives better results than the reconstruction model. This may be because prediction will always try to draw back to normality, whereas reconstruction works from pre-sight of an anomalous sequence. The comparison between the outputs from these two models are mentioned in more detail in the next section. We also compare performance results of two architectures with 12 layers and 8 layers. The deeper model gives better results on the three datasets.

As can be seen from Table 4.5, $\tau = 5$ give better results than $\tau = 2$ while increasing it to $\tau = 8$ does not improve the results. Moreover, data augmentation, which is named as aug2, using Gaussian image blurring, image sharpening, histogram equalization, crop and resize improves the results on UCSDPed1 and UCSDPed2. The number of training samples, which are double for UCSDPed1 and UCSDPed2 in this augmentation method (aug2), may help to train a more effective model. However, training the model on a larger training set takes a longer time.

We compare the performances of the convolutional encoder-decoder with skip connections and the convLSTM encoder-decoder for reconstruction and future prediction in Table 4.6. The prediction models overperform the reconstruction models for all architectures. As can be seen in this table, the results are the best with the use of the convolutional LSTM encoder-decoder for prediction.

Setup		EER/AUC (%)		
		UCSDPed1	UCSDPed2	CUHK-Avenue
Reconstruction	ConvLSTM	29.4/76	19.3/87	24.6/82.5
	2D Conv	29.6/77	20.3/86	27.2/79
Prediction	ConvLSTM	**24.4/81.8**	**13.1/92.8**	**22.7/84.6**
	2D Conv	27/79.1	17.7/89.3	24.8/82.6

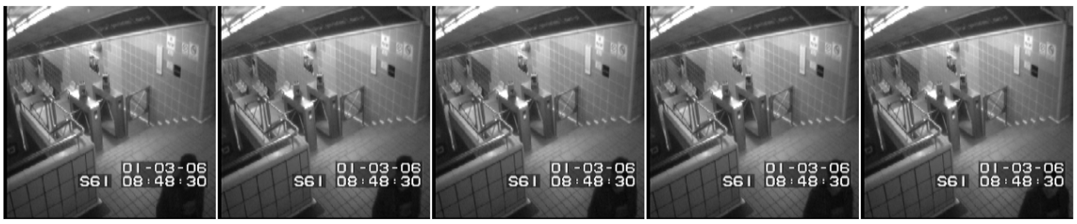
Table 4.6: Comparison of EER/AUC with different types of the encoder-decoders (the convolutional encoder-decoder and the convLSTM encoder-decoder) for reconstruction and prediction, aug2 is used for data augmentation, $\tau = 5$ is used.

Method	EER/AUC (%)				
	UCSDPed1	UCSDPed2	CUHK Avenue	Subway Entrance	Subway Exit
MDT [17]	25/81.1	25/85	-	-	-
SCL[16]	15/91.8	-	-	-	-
Conv-WTA[117]	14.8/91.6	**9.5/95**	26.5/81	-	-
AMDN[34]	16/92.1	17.1/90.8	-	-	
GAN [53]	-	15.6/93.5	-	-	-
Conv-AE [12]	27.9/81	21.7/90	25.1/70.2	26.0/**94.3**	9.9/80.7
ST-AE[13]	**12.5**/89.9	12.0/87.4	**20.7**/80.3	**23.7**/84.7	**9.5/94.0**
STAE-3D[14]	15.3/**92.3**	16.7/91.2	33.8/77.1	−	−
ConvLSTM-prediction-aug1	25.1/80.8	14.4/92.3	22.4/**84.8**	24.4/82.6	13.1/92
ConvLSTM-prediction-aug2	24.4/81.8	13.1/92.8	22.7/84.6	25.1/81.4	17.3/89.5

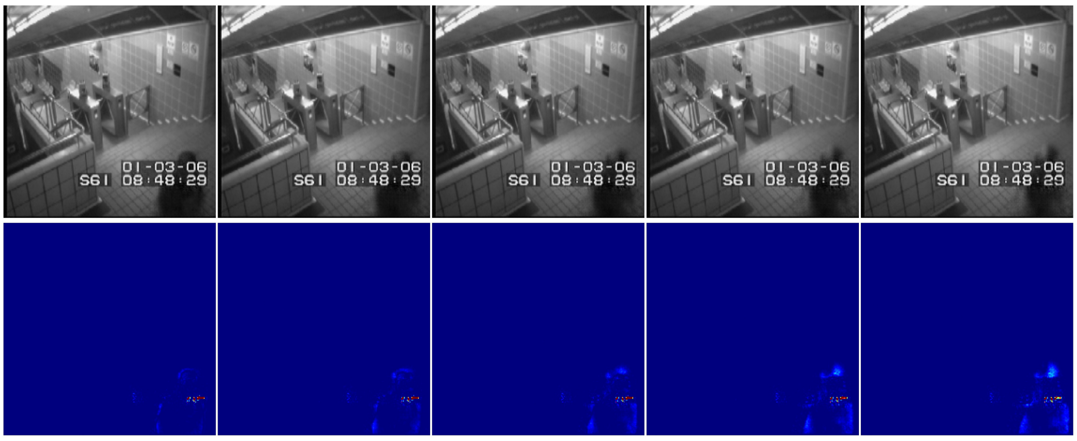
Table 4.7: Performance comparison with the state of the art.

We use the prediction models of the convolutional LSTM encoder-decoder with 12 layers and ($\tau = 5$) to compare with state of the art methods. Table 4.7 shows that the model trained for prediction performs comparably to state of the art results, including the three end-to-end deep learning methods at the bottom of the table (Conv-AE, ST-AE, STAE-3D). Performance on UCSDPed1 is relatively poor, whilst for CUHK Avenue, the AUC is 3% above the state of the art for which results are available. We have more false alarms in Subway Entrance/Exit dataset. We observe that the training data for both Entrance and Exit dataset is not good for representing normal events. The first 15 minutes of the Entrance dataset contains some anomalous events while the first 5 minutes of the Exit dataset contains one example of people normally exiting the gate. There are some variations of normal events in testing data that do not appear during training.

Moreover, we observe that video frames in the Subway dataset contains the time stamp in the right bottom corner of each frame. The prediction error for the timestamp affects the regularity score of the video sequence. For example, Figure 4.7 shows that big values of prediction error are obtained in the timestamp region when the second changes from 29 (in the input sequence) to 30 (in the target sequence). While we only focus on prediction error of objects in the scene, we mask out the errors for the timestamp area by setting all pixels in this area to zero. The results are significantly improved, as in Table 4.8.



Target sequence



Prediction sequence and pixel-wise prediction error

Figure 4.7: Prediction error in the timestamp area affects the regularity score. A blue-green-red color map shows error from low to high.

	EER/AUC (%)	
Method	Subway Entrance	Subway Exit
ConvLSTM-prediction-aug1		
Without masking	$24.4/82.6$	$13.1/92$
With masking	$\mathbf{15.9/90.2}$	$\mathbf{8/95}$
ConvLSTM-prediction-aug2		
Without masking	$25.1/81.4$	$17.3/89.5$
With masking	$\mathbf{17.2/89.1}$	$\mathbf{10.9/93.7}$

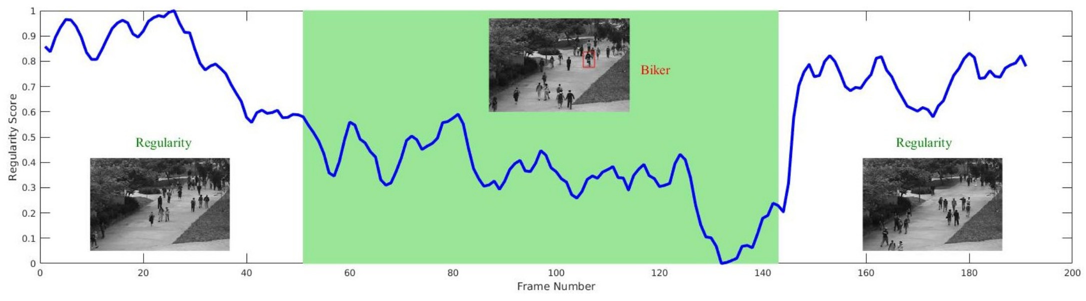
Table 4.8: Performance comparison in Subway Entrance/Exit datasets with and without masking the timestamp.

Figure 4.8, 4.9, 4.10, 6.5 and 6.6 show the regularity score derived from prediction error of sequences in some videos of the UCSD, CUHK Avenue and Subway datasets. In these figures, green shaded regions represent ground-truth abnormal frames. Figure 4.8 shows the regularity score for five video sequences in UCSDPed1. The regularity score is low when a biker or a car appears in the scene. However, normal frames are falsely detected as anomalies in some cases. For example, Figure 4.8 (d) and (e) show that the crowd density of the scene affects the regularity score. Complex movements of a group of people results in a low regularity score. Moreover, the camera shaking in video sequence #17 also affects the score.

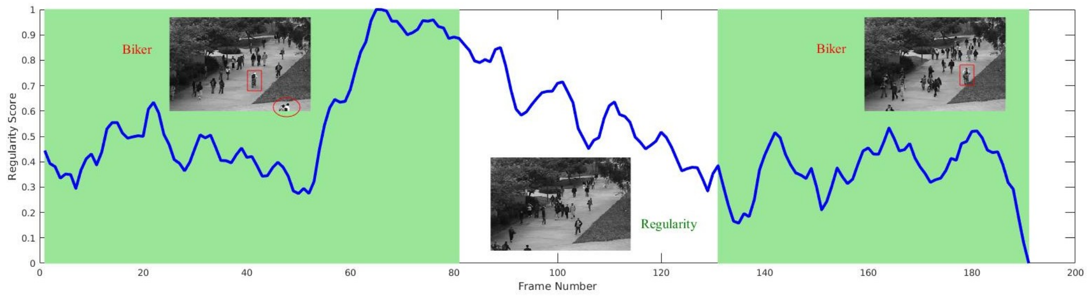
As shown in Figure 4.9, anomalies in UCSDPed2 are captured better with low regularity scores. In the CUHK Avenue dataset, low regularity scores are obtained in the video segments which contain throwing, running and someone moving in the opposite direction as illustrated in Figure 4.10. However, since the horizontal movement of a person in the close-field of the scene rarely appears in training data, it results in low regularity score for the test sequence (for example, sequence #7 and #12 in Figure 4.10 (b) and (d), respectively).

Figure 6.5 and Figure 6.6 shows the regularity score of video sequences in Subway Entrance and Exit datasets. The regularity score is fluctuated significantly in the video segments which do not contain any motion. By masking the timestamp, the regularity score becomes more robust to capture anomalies, where low scores are obtained when abnormal events such as no payment, running, loitering and wrong direction appear in the scene.

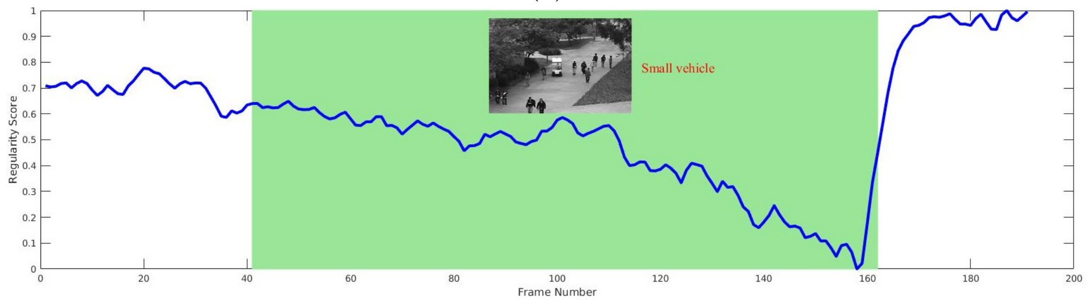
More visualizations of regularity score are included in Annex (Section 6.3).



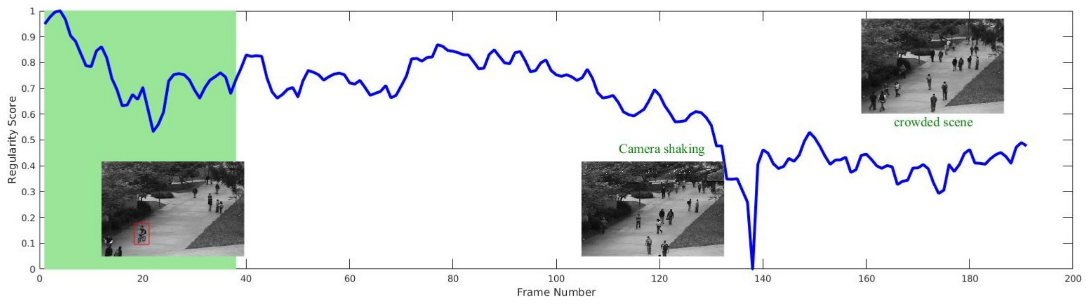
(a)



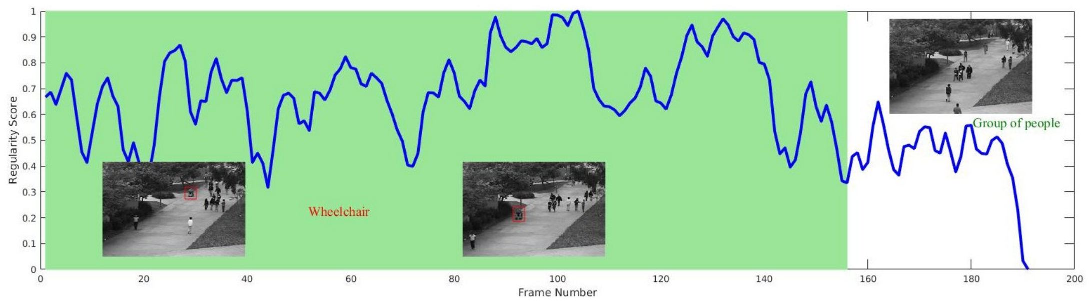
(b)



(c)



(d)



(e)

Figure 4.8: Regularity score of video sequence $\#1, 5, 24, 17, 23$ (from top to bottom) of UCSDPed1 dataset.

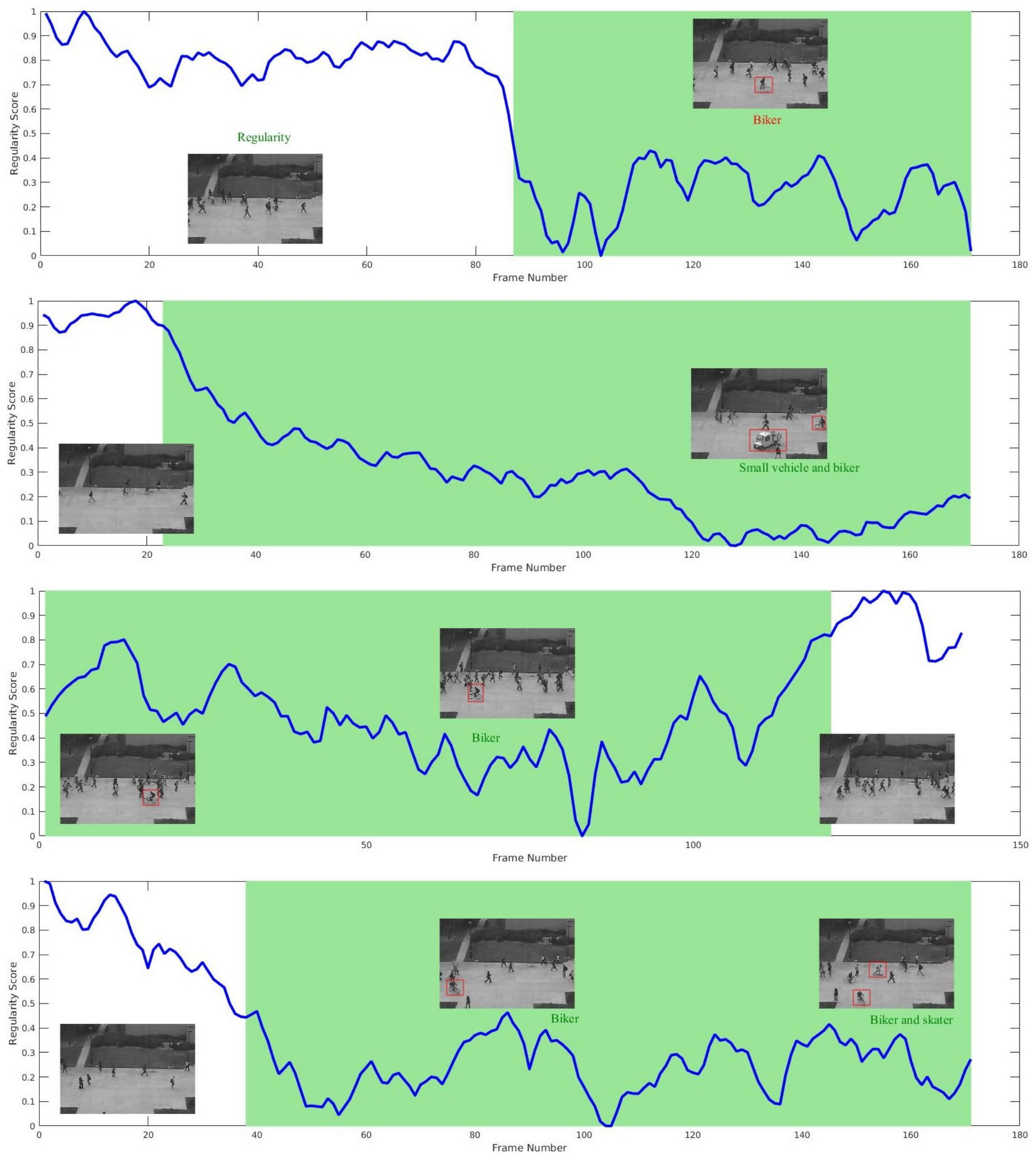
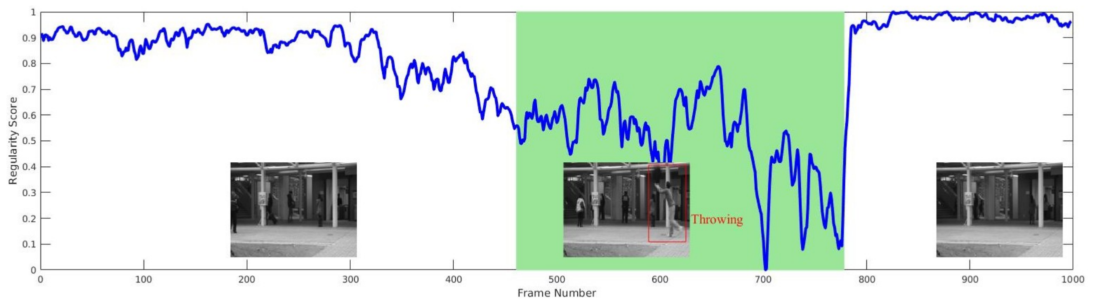
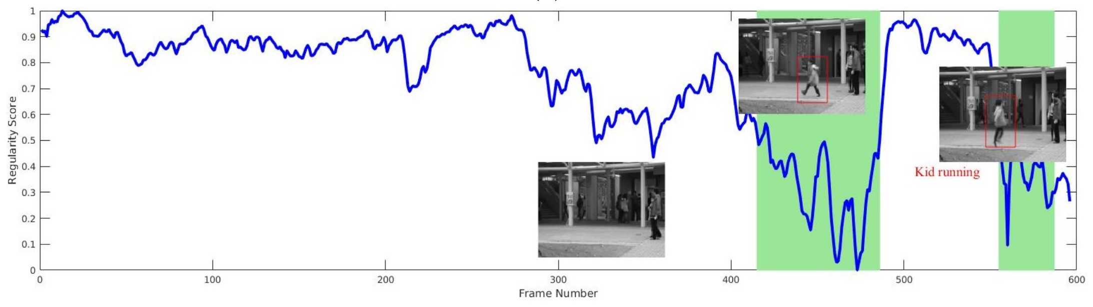


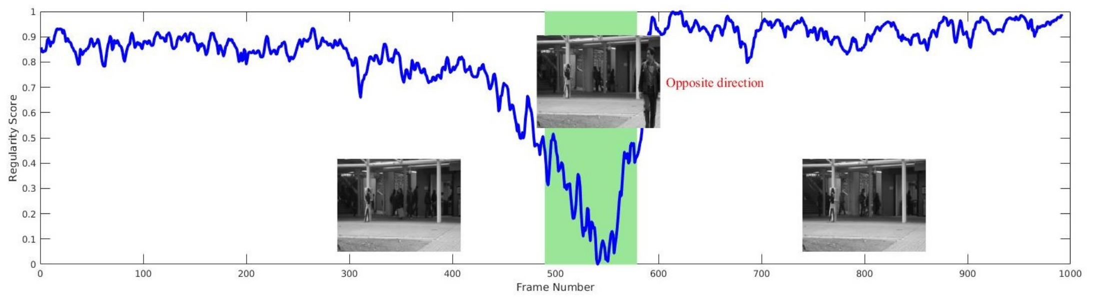
Figure 4.9: Regularity score of video sequence $\#2, 4, 5, 7$ (from top to bottom) of UCSDPed2 dataset.



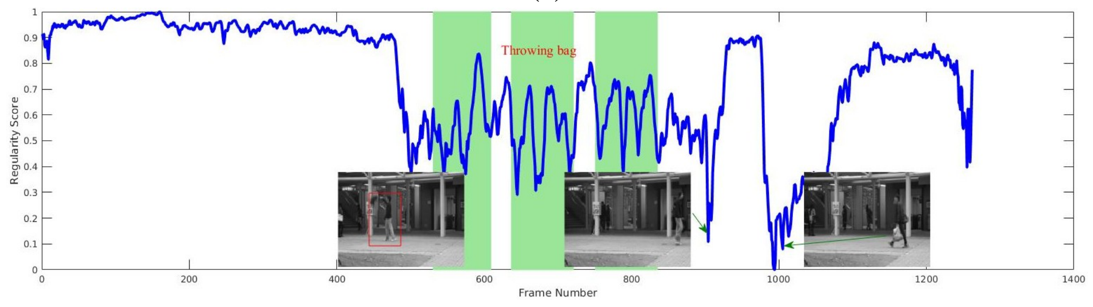
(a)



(b)

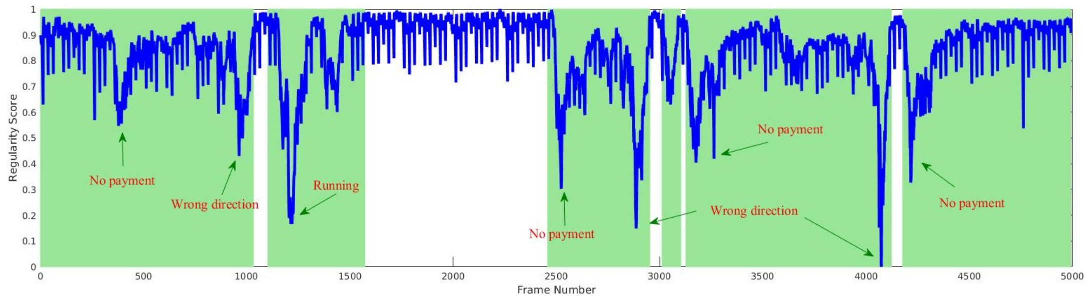


(c)

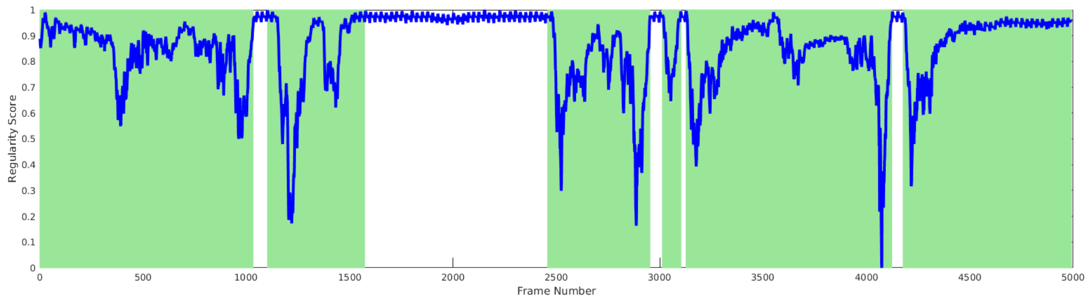


(d)

Figure 4.10: Regularity score of each sequence of video sequence $\#5, 7, 15, 12$ (from top to bottom) of CUHK Avenue dataset.

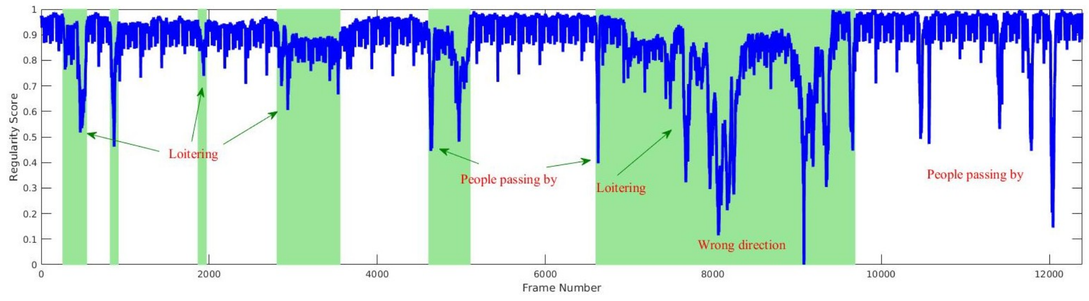


(a) Without masking the timestamp

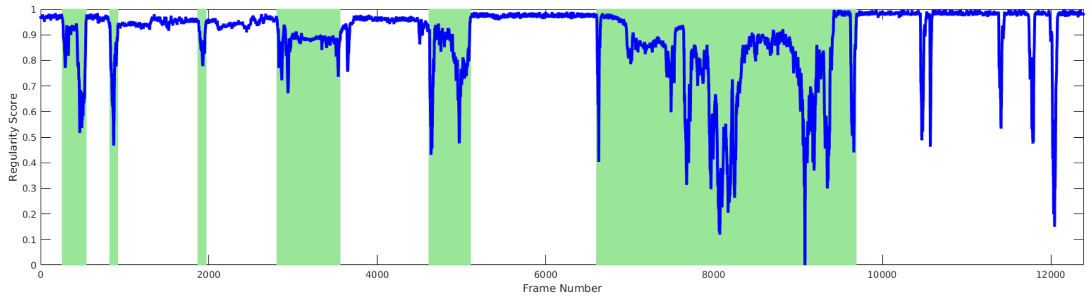


(b) With masking the timestamp

Figure 4.11: Regularity score of frames $\#115,000 - 120,000$ of Subway entrance dataset.



(a) Without masking the timestamp



(b) With masking the timestamp

Figure 4.12: Regularity score of frames $\#52,500 - 64,000$ of Subway exit dataset (without and with masking the timestamp).

4.5.3 Reconstruction and Prediction

In this section, we visualize the outputs of the convolutional LSTM encoder-decoders that are used for reconstruction and future prediction. We use the models with 12 layers, $\tau = 5$ to produce the outputs. In order to visualize the outputs and compare the errors of two models, we feed an input volume of 5 frames $\mathcal{F}_{t-4}, ..., \mathcal{F}_t$ to the prediction model to predict the next 5 frames $\mathcal{F}_{t+1}, ..., \mathcal{F}_{t+5}$; and we feed an input volume $\mathcal{F}_{t+1}, ...\mathcal{F}_{t+4}, \mathcal{F}_{t+5}$ to the reconstruction model to reconstruct 5 frames $\mathcal{F}_{t+5}, \mathcal{F}_{t+4}..., \mathcal{F}_{t+1}$. Basically, the target volumes are the same but with reverse order.

Figure 4.13, 4.14, 4.15 and 4.16 show the reconstruction and prediction on sample irregular frames of UCSDPed1, UCSDPed2 and CUHK Avenue dataset in which anomaly is a car, a wheelchair, a biker and a running person, respectively. In these figures, the first row shows the target volume, the second and the fourth rows illustrate the prediction and the reconstruction, respectively. The third and the fifth rows show the pixel-wise prediction error and the reconstruction error between each frame in the output sequence comparing to the target sequence, respectively. The final error $e(t)$ is calculated by summing all pixel-wise errors. We use a blue-green-red color map to show the degree of anomaly in which blue is normal and red is abnormal.

As shown in the figures, the reconstruction error is usually smaller than prediction error. This observation is reasonable because playing back the input sequence from the learned features in hidden states is easier than predicting the unknown sequence in the future. We trained our model only on normal data and we expect that only normal appearances and dynamics are captured and have better reconstruction/prediction in comparison with anomaly. Using reconstruction models, both normal and abnormal objects are reconstructed fairly well. The pedestrian is better reconstructed but car, biker and running child are also well reconstructed. In the prediction model, the estimation of pedestrian is worse than its reconstruction. Moreover, the prediction of the biker looks similar to pedestrian while the prediction of the car and a running child disappears or becomes increasingly blurred at each time step into the future.

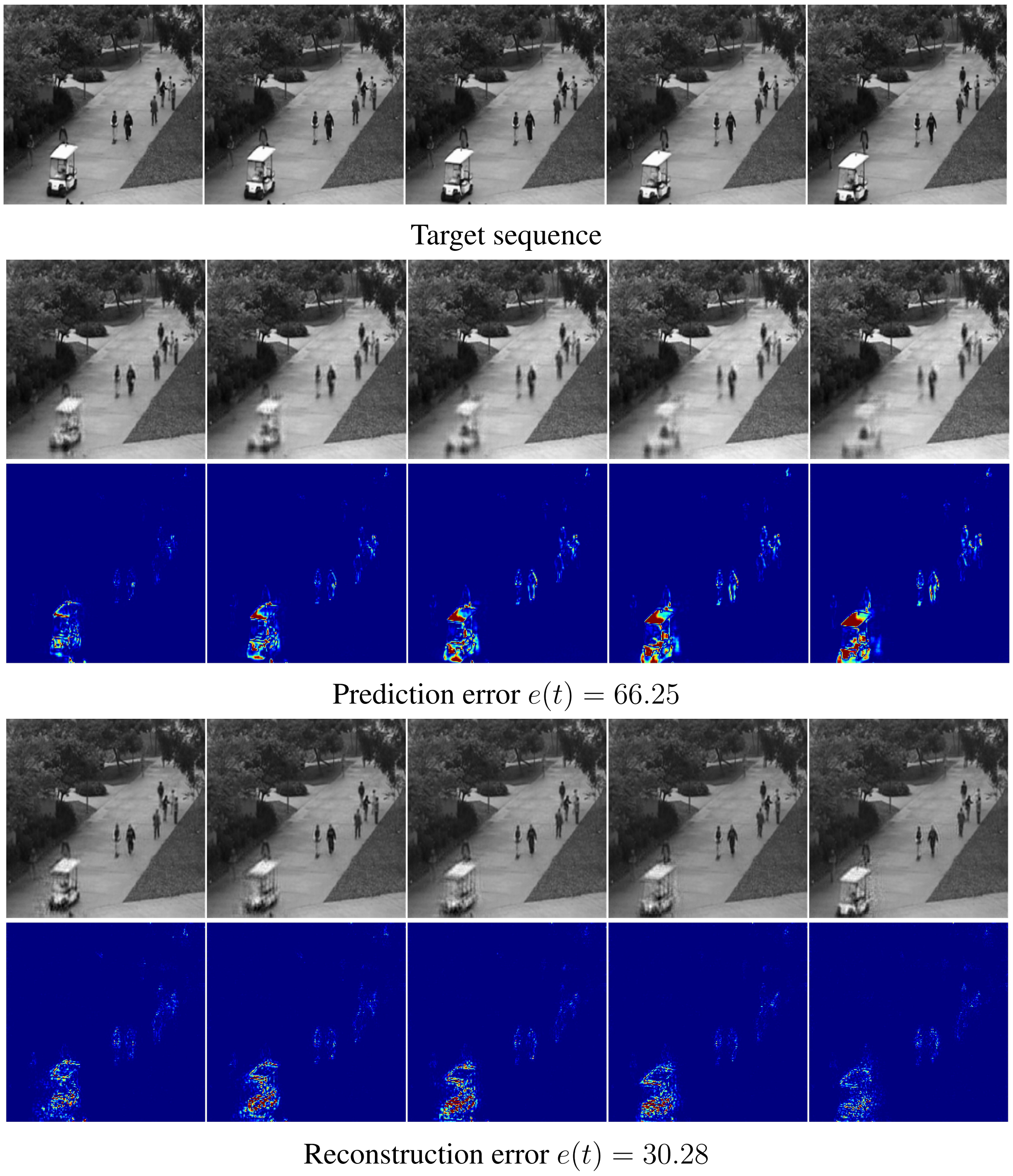

Target sequence
Prediction error $e(t) = 66.25$
Reconstruction error $e(t) = 30.28$

Figure 4.13: Reconstruction and prediction on sample irregular frames of UCSDPed1 which contains a car. Best viewed in color.

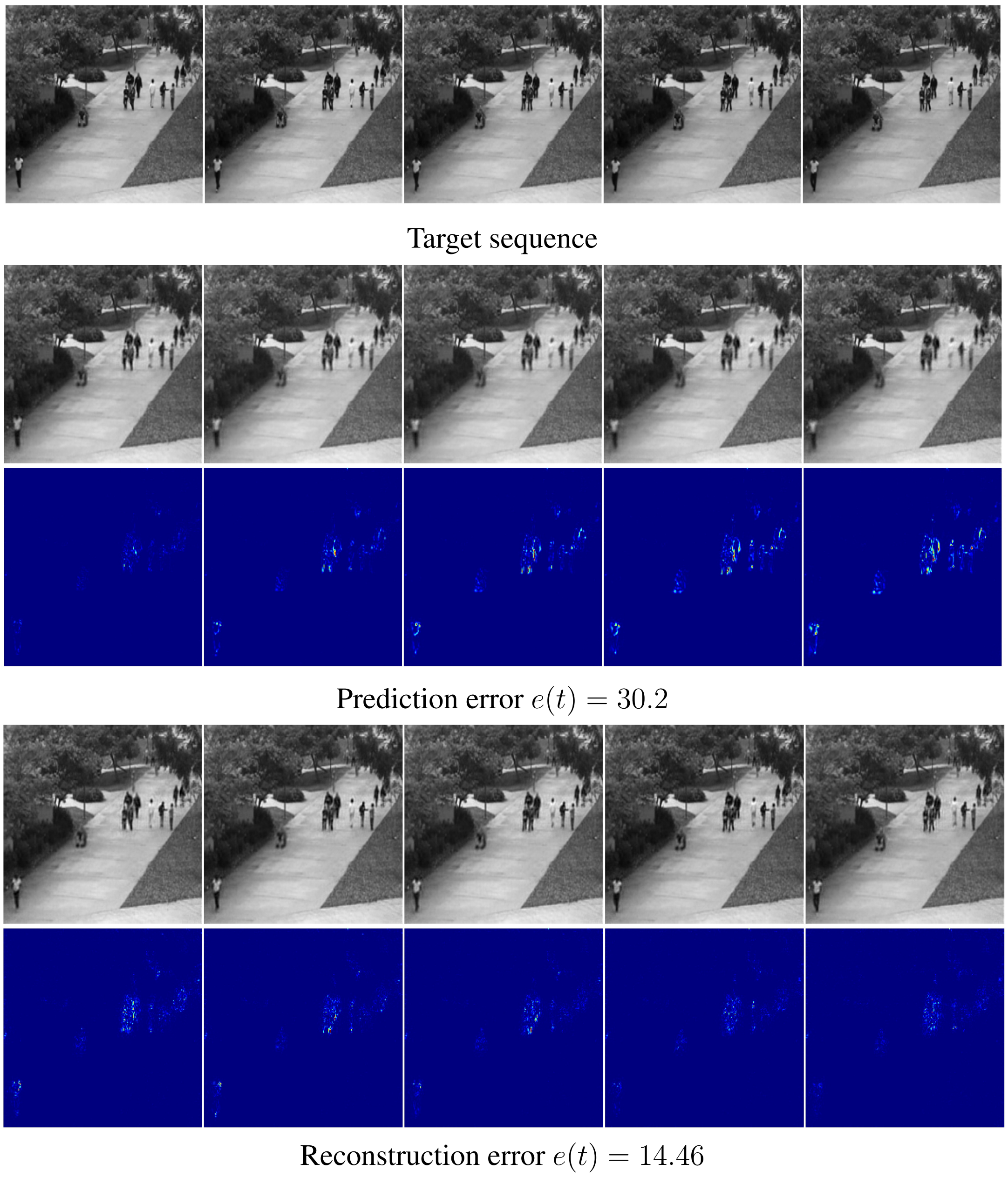

Target sequence
Prediction error $e(t) = 30.2$
Reconstruction error $e(t) = 14.46$

Figure 4.14: Reconstruction and prediction on sample irregular frames of UCSDPed1 which contains a wheelchair. Best viewed in color.

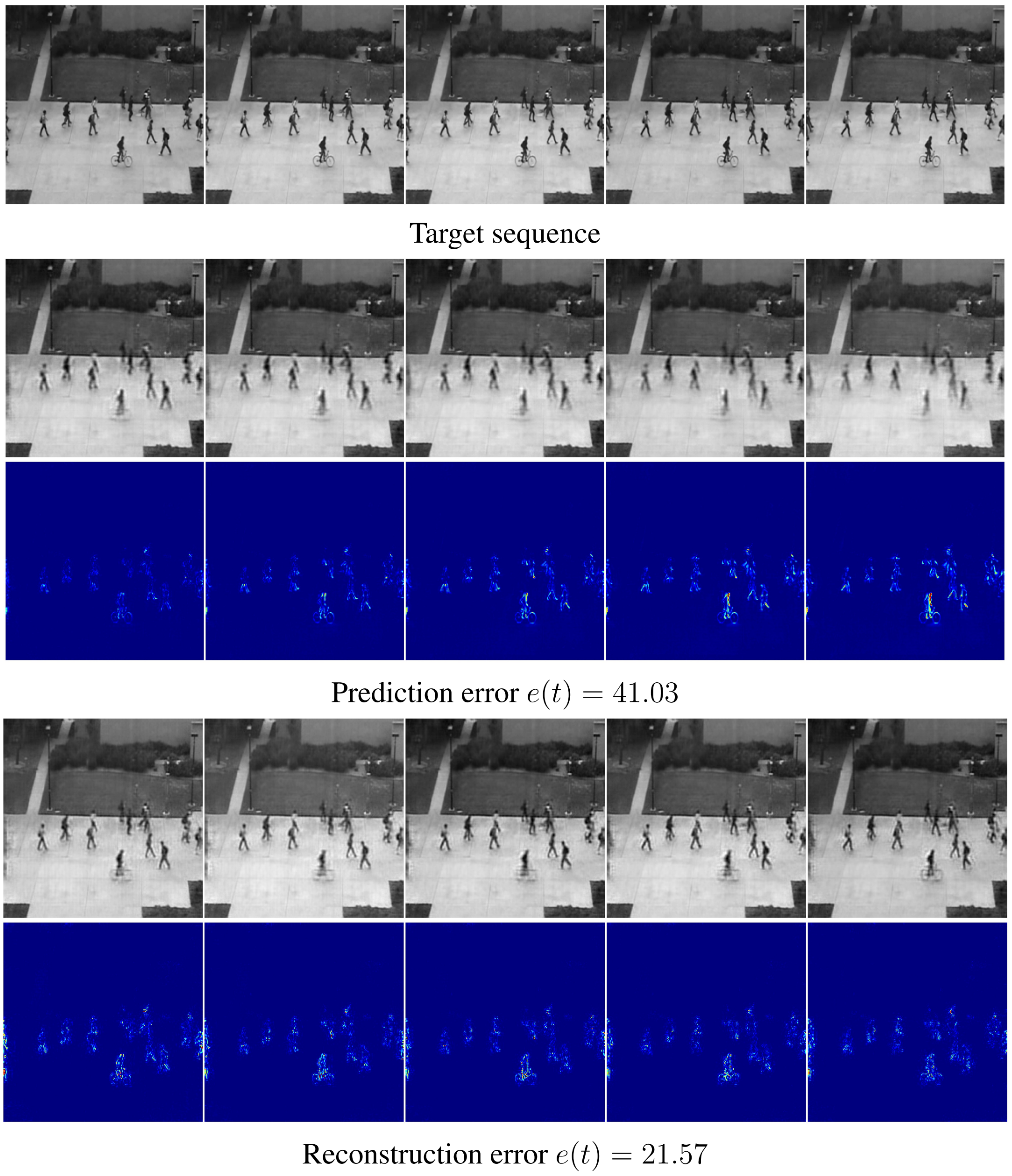
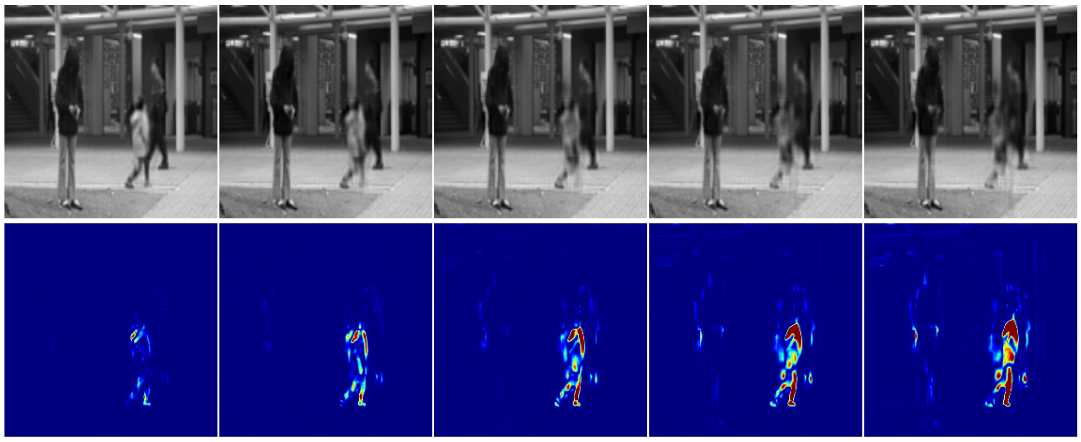
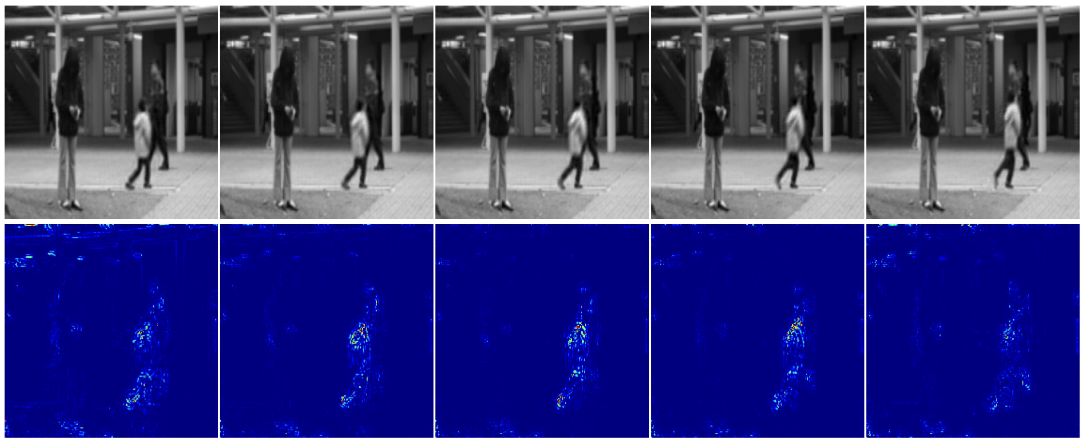

Target sequence
Prediction error $e(t) = 41.03$
Reconstruction error $e(t) = 21.57$

Figure 4.15: Reconstruction and prediction on sample irregular frames of UCSDPed2 which contains a biker. Best viewed in color.

Target sequence



Prediction error $e(t) = 59.81$



Reconstruction error $e(t) = 11$

Figure 4.16: Reconstruction and prediction on sample irregular frames of CUHK Avenue which contains a running person. Best viewed in color.

4.5.4 Number of model parameters and tesing time

In this section, we compare the number of model parameters for the method against different end-to-end trainable models in the state of the art as shown in Table 4.9. We only count the layer which has parameters in # layers column.

Method	# layers	# parameters
Conv-AE [12]	6	$8,382,464$
ST-AE[13]	7	$1,073,384$
STAE-3D[14]	8	$541,728$
Ours	12	$845,698$

Table 4.9: Comparison on number of parameters.

We also present a run-time on CPU (Intel core i7-4790) and GPU (GeForce GTX TITAN X) in Table 4.10. The convolutional LSTM predictor with 12 layers and $\tau = 5$ is used to calculate the run-time. Due to the capacity of parallel processing of convolution operations on the GPU, the run time is significant faster than testing with only a CPU. In the table, the processing time refers to the time for resizing video frames and stacking them into a video volume. The prediction refers to the time for the model to output the estimation and calculate the prediction error.

	Processing	Prediction
Without GPU	0.00154	2.7907
With GPU	0.00154	$0.0132(\sim 75fps)$

Table 4.10: Testing time (second/frame) without and with GPU.

4.6 Conclusion

We have adapted a state of the art predictive encoder-decoder deep network to detect abnormal events in video. We evaluate detection performance using both sequence prediction and reconstruction, and show that prediction gives superior anomaly detection performance over the reconstruction. For the prediction model, we obtain competitive performance to state of the art methods on three standard datasets. We evaluate performance with two different numbers of layers and two data augmentation methods. We also compare performance of the convolutional LSTM encoder-decoder and the convolutional encoder-decoder skip connections. Finally, we evaluate performance across different prediction windows, encompassing varying levels of motion complexity.

Chapter 5

Conclusion and Future Work

This thesis investigated the problem of anomaly detection in crowded scenes. We presented our approach to this problem in two different parts of the thesis, (i) using the convolutional autoencoder to learn motion feature representations and a one-class SVM to learn the model of normality, (ii) training the convolutional encoder-decoders to learn normal spatio-temporal dynamics and the use of prediction error for anomaly detection. Each of these novel frameworks are briefly described below.

In Chapter 3 we presented a method for video anomaly detection using a convolutional Winner-Take-All (WTA) autoencoder and a one-class SVM (OCSVM). We provided the full literature review of the feature descriptors that were used in this area in Section 2.2. However, these descriptors are hand-crafted. We also reviewed the use of deep models to extract feature representations for anomaly detection in Section 2.3.2. However, these models are either pretrained models on a large-scale image dataset, which were not designed for learning motions, or models trained with fully connected layers, which could not learn spatial information. We therefore presented our method using a convolutional autoencoder with a WTA step to produce a spare representation. In order to evaluate the efficiency of this spatial sparsity step, we also presented a convolutional autoencoder, replacing the WTA step with a max-pooling layer to learn compressed features.

In Chapter 4 we presented our method based on the use of the convolutional encoder-decoders for anomaly detection. Our method in Chapter 3 required one-class SVM to be learned on the features extracted on training data, thus the whole system could not be trained end-to-end. Therefore, we presented a convolutional LSTM encoder-decoder to learn normal spatio-temporal dynamics from sequences of successive frames on training data which contain mostly normal events. The encoder-decoder was trained to minimise the error between network's outputs and targets, thus the error should be small for a normal test volume. We used the regularity score, which was derived from a prediction error or a reconstruction error, to classify a test volume as a normality or an anomaly.

5.1 Contributions

5.1.1 Convolutional Winner-Take-All autoencoder

We introduced a method for video anomaly detection that exploits the use of a convolutional autoencoder on foreground optical flow patches, coupled with a spatial winner-take-all step, to learn shift-invariant and generic flow features. In addition, a max-pooling layer and a temporal step are used following the trained encoder to reduce the dimensionality of the coding features and to form a final smoothed feature representation. We also employ local normality modelling in which the field of view is divided into regions and a one-class SVM is independently used within each region. This local modelling method helps to detect contextual anomalies, and anomalies at different scales through the scene.

Experimental results in Chapter 3 shows that using a convolutional autoencoder with winner-take-all step improves the performance over the use of a convolutional encoder with a max-pooling layer. Moreover, our method outperforms state-of-the-art approaches on two challenging datasets.

5.1.2 Convolutional Long Short-Term Memory

We introduced an end-to-end framework that learns normal spatio-temporal dynamics using sequence-to-sequence encoder-decoders for prediction and reconstruction. This is done by interleaving LSTM based RNNs between convolutional layers to encode temporal information on hierarchical spatial representations from low-level to high-level. To evaluate the efficiency of LSTMs in learning temporal dynamics, we also introduced a convolutional encoder-decoder with skip connections.

Beside the use of reconstruction error, we proposed to use prediction error for anomaly detection. We train the encoder-decoders for both reconstruction and future prediction, and show that prediction error gives superior performance than the use of reconstruction.

Based on the evaluations in Chapter 4, our prediction models give competitive performance to state of the art methods on three challenging datasets. Moreover, the convolutional

LSTM encoder-decoder outperforms the convolutional encoder-decoder with skip connections.

The convolutional LSTM predictor improves the results of using a convolutional WTA feature extractor and a global OCSVM on UCSDPed1 and CUHK Avenue datasets. However, it gives worse results on UCSDPed2. This may be due to the lack of training volumes in this subset, which is three times less than the number of training volumes on UCSDPed1 and five times less than the number of training volumes on CUHK Avenue. Another improvement of the convolutional LSTM encoder-decoder is the running time which is speeded up around 50 times. The encoder-decoder learns the motion dynamics from a stack of successive frames, thus there is no need to compute optical flow in a prior step.

5.2 Limitations

There are a few limitations associated with the proposed methods.

- The work in Chapter 3 assumes that anomalies are only found where there is non-zero optical flow in the image plane. In particularly, we considered patches with accumulated optical flow squared magnitude above a fixed threshold are foreground. Therefore, the threshold value affects the detection results, a big value may remove abnormal patterns while a small value keeps background patterns as foreground.

- In Chapter 3, we employed a temporal smoothing in the convolutional WTA feature extractor to extract a robust representation. Since we used a temporal window of $\tau = 5$, the method cannot capture a longer-time consistency of an anomaly. Specifically, there are still some false and missing detections due to the occlusion of moving objects.

- Hyper-parameters for deep models such as kernel size, number of filters, number of layers in our encoder-decoder architectures were chosen based on the common use of them in state of the art architectures for classification tasks and anomaly detection tasks. We trained our models with different optimization methods such as stochastic gradient descent (SGD) or Adam. However, we evaluated the use of the optimization methods on one architecture of an autoencoder and then applied it on other architectures, supposing the method works well with these architectures. Moreover, we manually changed the value of learning rate, decreased it 10 times if training error increased.

- In Chapter 4, the encoder-decoders for reconstruction identify anomalies by a poor reconstruction of objects that have never appeared in the training data. However, the encoder-decoders are able to reconstruct anomalies fairly accurately even when know nothing about the anomalies.

5.3 Future Work

Various improvements and extensions may be applied to each of the proposed frameworks to solve the above limitations. In this section we present several directions as part of future work.

An abnormal event is supposed to appear in the video in a continuous spatio-temporal locations. However, our framework in Chapter 4 detected foreground patches separately as normalities or anomalies. It would be of benefit to track abnormal patches over longer time duration. This would help to remove false positives and fill missing detections.

Currently, our frameworks only perform on either optical flow (in Chapter 3) or raw pixel data (in Chapter 4). It would be interesting to add an appearance channel alongside the optical flow channel. Fusion can be done by concatenating optical flow and appearance to form input data for a network or concatenating their feature tensors to learn spatio-temporal features. Moreover, the weighted sum of abnormal scores of two separate networks for optical flow and appearance can be used to consider anomalies in appearance and motion.

It would be of benefit to run an exhaustive grid search to optimize and tune the hyper-parameters of our deep models. We ran grid search to find parameters for training a one-class SVM. However we did not do it for every change we make to the method, for example the change in a patch size, pooling size or length of temporal smoothing window. It would be interesting to see the extent to which re-tuning hyper-parameters affects the final result.

It would be interesting to replace deconvolutional layers in our encoder-decoders in Chapter 4 with other up-scaling layers such as a sub-pixel convolutional neural network layer (also called PixelShufle) [118]. Instead of padding zeros in between pixels as in a deconvolutional layer, the sub-pixel convolution layer uses regular convolutional layers followed by a specific type of image reshaping which is called a phase shift.

Encoder-decoders for reconstruction in Chapter 4 performed more poorly than encoder-decoders for prediction. Through qualitative analysis, we observed that the encoder-decoder reconstructs both normalities and anomalies fairly well. It would be interesting to extend the existing encoder-decoder for reconstruction (i.e. an autoencoder) using the following ideas. Firstly, we could add noise (e.g. Gaussian noise) to the input volumes and force the autoencoder to reconstruct clean inputs. The autoencoder would consider an abnormal object as noise and reconstruct it poorly. Secondly, it would be interesting to use negative samples (i.e, anomalies) and force the autoencoder to produce bad reconstructions for these samples. By doing this, the autoencoder reconstructs normal data with small error but reconstructs abnormal data with larger error. Since labelled data is not available for anomaly detection, it would be interesting to consider the use of synthetic abnormal data, for example in the case of the UCSD dataset which only contains pedestrians, scenes containing non-pedestrians may be used as abnormal data. However, this method requires negative samples to be collected manually.

Finally, we could adapt the idea of having a generator producing the contrastive samples in Generative Adversarial Networks [119] into our existing encoder-decoders, which is shown in Figure 5.1.

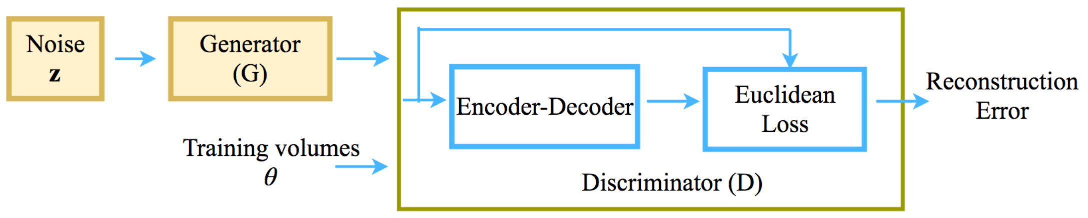


Figure 5.1: A proposed encoder-decoder with the presence of negative samples from the generator .

The generator G is trained to produce a video volume $G(\mathbf{z})$ from a random vector $\mathbf{z}$, which is sampled from a Gaussian distribution. The discriminator takes either real (normal) volumes or generated volumes and reconstructs them, attributing low error to the training volumes and higher error to the generated ones. The generator (G) and discriminator (D) can be trained to minimise the following loss functions:

$$\begin{aligned}\mathcal{L}_D(\boldsymbol{\theta}, \mathbf{z}) &= D(\boldsymbol{\theta}) + \max(0, m - D(G(\mathbf{z}))) \\ \mathcal{L}_G(\mathbf{z}) &= D(G(\mathbf{z}))\end{aligned} \tag{5.1}$$

where $D(.)$ is reconstruction error, m is a positive margin.

5.4 Closing Remarks

This thesis has introduced frameworks where deep convolutional networks are used to learn spatio-temporal dynamics on optical flow fields or on raw data for crowded scenes.

Based on quantitative and qualitative results, we experimentally validated the use of the deep encoder-decoder in the problem without labelled data. The proposed methods work on different datasets, which requires less prior knowledge of the target datasets. While this work is a small but important step towards exploiting deep feature learning for the anomaly detection problem, we hope to see a substantial improvement in this approach where a normal manifold is learned automatically and effectively on normal data in any domain.

Chapter 6

Annex

6.1 Convolutional WTA autoencoder for anomaly detection

6.1.1 One-class SVM kernels

Table 6.1 shows results when two kernels are used for one-class SVM. In these experiments, Min-Max scaling is used for normalizing features of training patches.

- Gaussian kernel: $k(\mathbf{d}_i, \mathbf{d}) = e^{-\gamma \|\mathbf{d}_i - \mathbf{d}\|^2}$
- Linear kernel: $k(\mathbf{d}_i, \mathbf{d}) = \mathbf{d}_i' * \mathbf{d}$

OCSVM kernel	Frame level (%)	Pixel level (%)	Running time (s)	
	EER/AUC	EER/AUC	Feature extraction	OCSVM
Gaussian kernel	$\mathbf{8.7/97}$	$\mathbf{16.6/89.8}$	0.0834	0.0231
Linear kernel	$44.1/57.5$	–	0.0771	6.2693

Table 6.1: Performance comparison on UCSDPed2 with different kernels for OCSVM.

6.1.2 Normalization as a preprocessing step for OCSVM.

Table 6.2 shows performance results on UCSDPed2 with two different normalization methods for OCSVM. Gaussian kernel is used in all experiments.

- Min-Max scaling:

$$\mathbf{d}_{i-norm} = \frac{\mathbf{d}_i - \min\limits_{j=1:M} \mathbf{d}_j}{\max\limits_{j=1:M} \mathbf{d}_j - \min\limits_{j=1:M} \mathbf{d}_j} \tag{6.1}$$

- Standardization (Z-score normalization):

$$\mathbf{d}_{i-norm} = \frac{\mathbf{d}_i - \underset{j=1:M}{\text{mean}} \mathbf{d}_j}{\underset{j=1:M}{\text{std}}\ \mathbf{d}_j} \tag{6.2}$$

OCSVM kernel	Frame level (%)	Pixel level (%)
	EER/AUC	EER/AUC
Min-Max scaling ($\gamma = 2^{-7}, \nu = 2^{-9}$)	$\mathbf{8.7/97}$	$\mathbf{16.6/89.8}$
Standardization ($\gamma = 2^{-11}, \nu = 2^{-9}$)	$10.8/95.8$	$19/85.4$

Table 6.2: Performance comparison on UCSDPed2 with different normalization methods for OCSVM.

6.1.3 A convolutional WTA autoencoder with different number of convolutional layers.

In this section, we present EER/AUC results obtained with different number of convolutional layers in the convolutional WTA autoencoder. The network with three convolutional layers was described in Chapter 3. In order to create a deeper network (with six convolutional layers), we add a max-pooling layer and three convolutional layers on top of the network used in Chapter 3. The architecture of the deeper network is: 128conv5-128conv5-128conv5-pool2-128conv3-128conv3-128conv3-WTA-128deconv21. The first three layers are initialized using the trained model in Chapter 3. The remained convolutional layers are initialised using He et al. method [87].

Number of layers	**UCSDPed1** Conv-WTA-OCSVM[6×9] Frame level % EER/AUC	**UCSDPed1** Conv-WTA-OCSVM[6×9] Pixel level % EER/AUC	**UCSDPed2** Conv-WTA-OCSVM[1×1] Frame level % EER/AUC	**UCSDPed2** Conv-WTA-OCSVM[1×1] Pixel level % EER/AUC
3 layers	**15.4/91.4**	**33.9/66.9**	**8.72/97**	**16.6/89.8**
6 layers	18.7/89.5	38.3/60.4	10.5/95.3	18.6/85.3

Table 6.3: Performance comparison on UCSDPed1 and UCSDPed2 with different network's architectures.

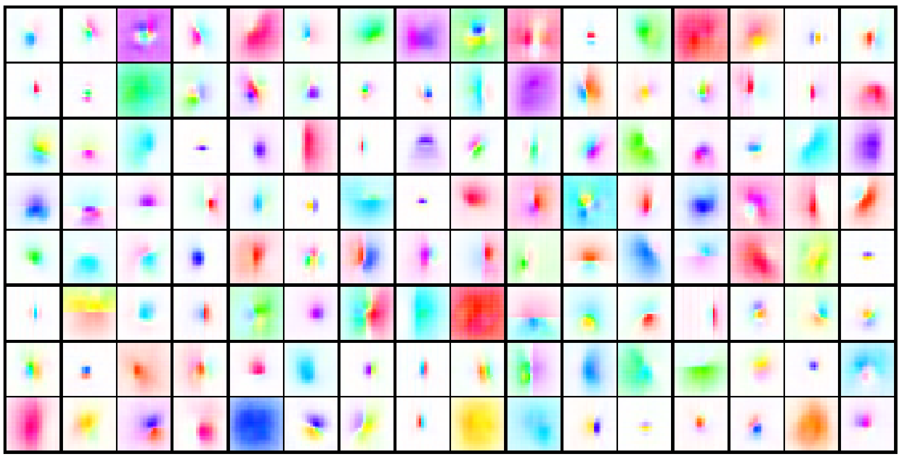


Figure 6.1: Learned deconvolutional filters of the Conv-WTA autoencoder with 6 convolutional layers trained on optical flow foreground patches (UCSD dataset).

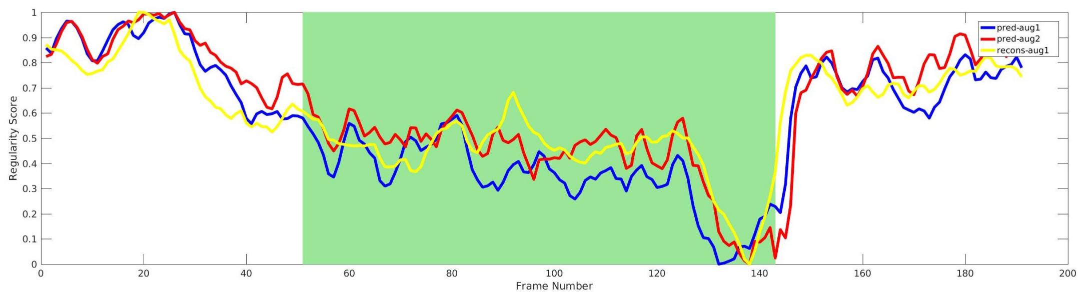
Table 6.3 shows that the model with three convolutional layers gives better results in both UCSDPed1 and UCSDPed2. We used deeper network to learn higher level feature representations (for example, motions of pedestrians or motion of upper part of pedestrian's body). However, it seems that our model with 6 convolutional layers does not learn these features effectively (Figure 6.1). Most of the deconvolutional filters capture low-level feature representations.

6.2 Convolutional Long Short-Term Memory for anomaly detection

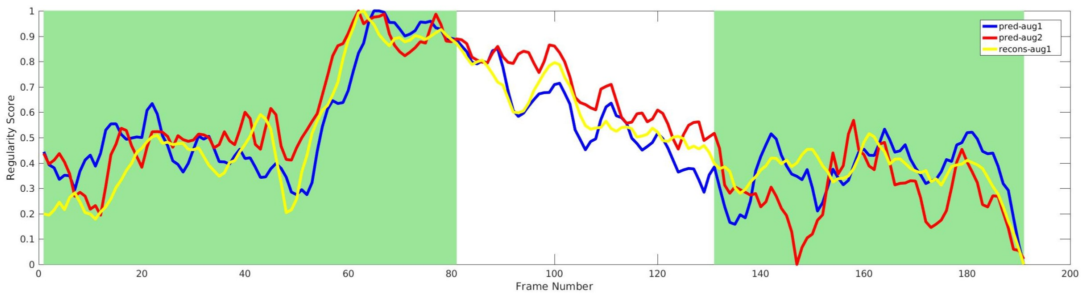
6.2.1 Regularity score of different models.

In this section, we visualize regularity scores of prediction model (with two augmentation methods aug1, aug2) and reconstruction model (with aug1).

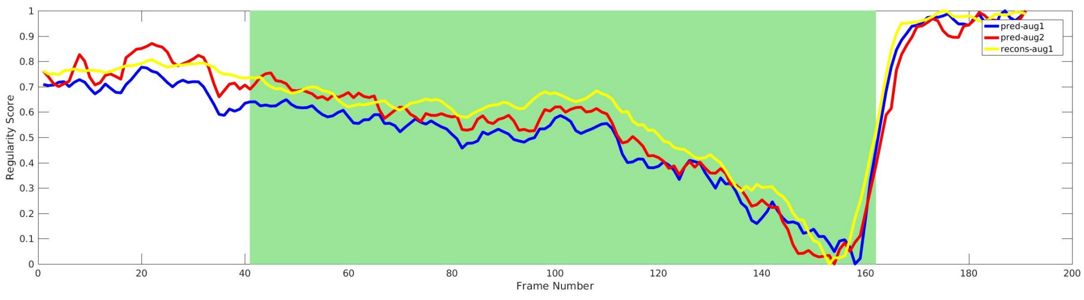
Figure 6.2, 6.3 and 6.4 show that regularity score derived from prediction error captures anomalies better than the one from reconstruction error. The regularity score are low for anomalies.



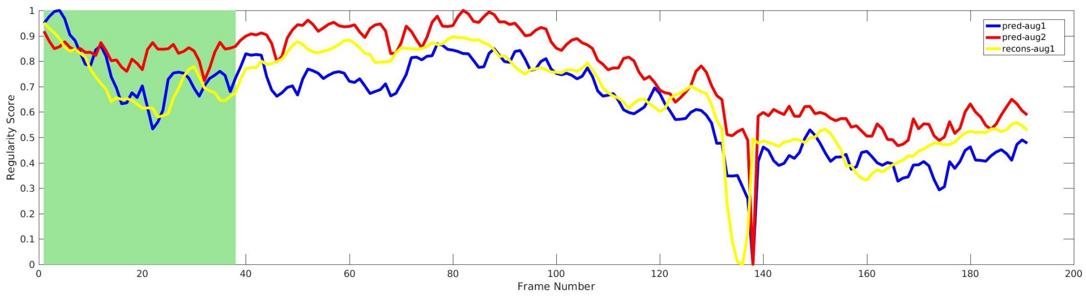
(a)



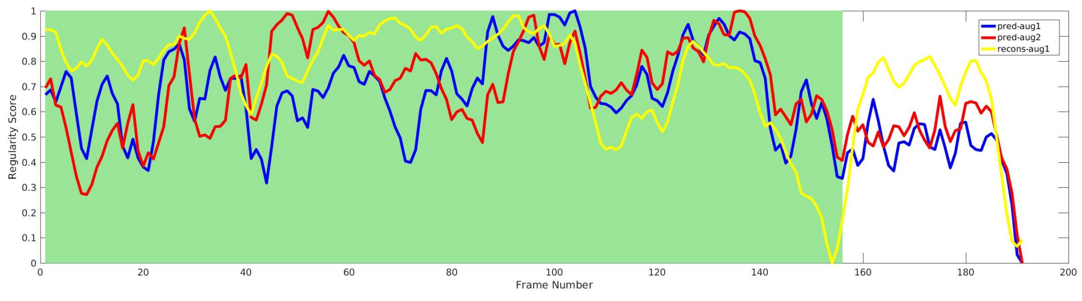
(b)



(c)



(d)



(e)

Figure 6.2: Regularity score of video sequence $\#1, 5, 24, 17, 23$ (from top to bottom) of UCSDPed1 dataset.

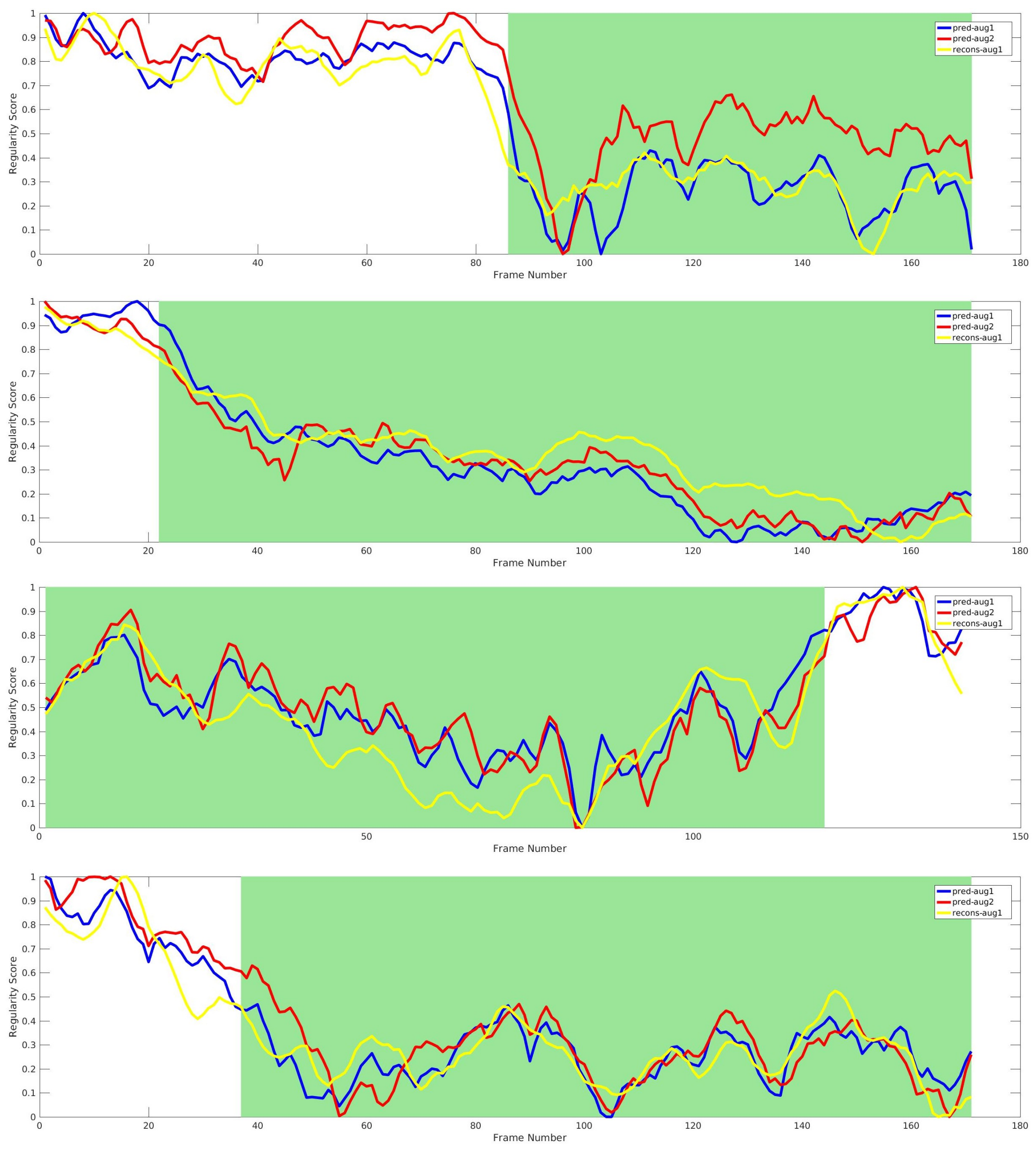
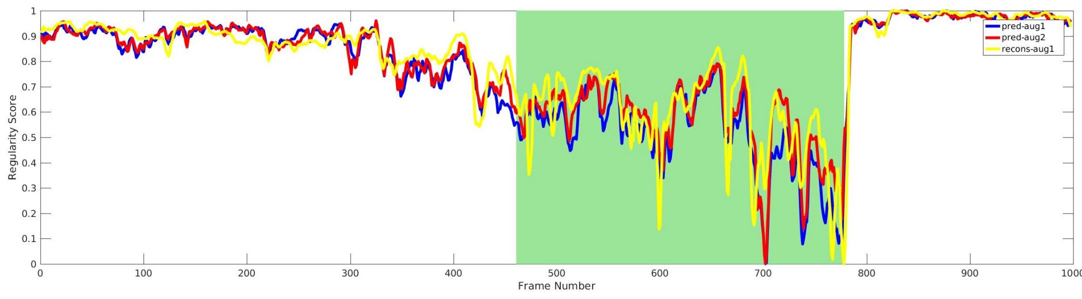
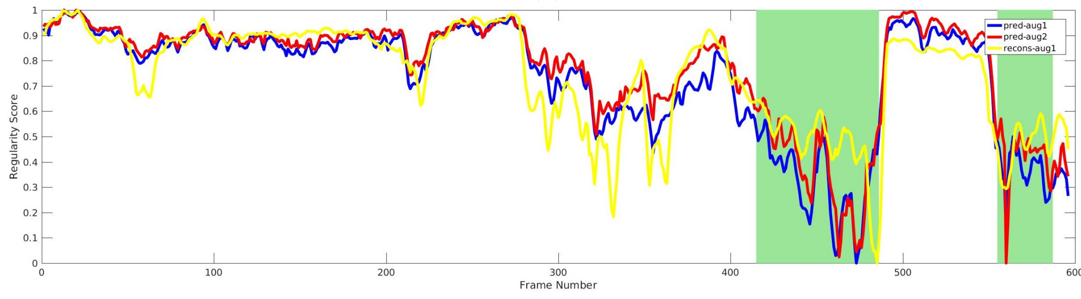


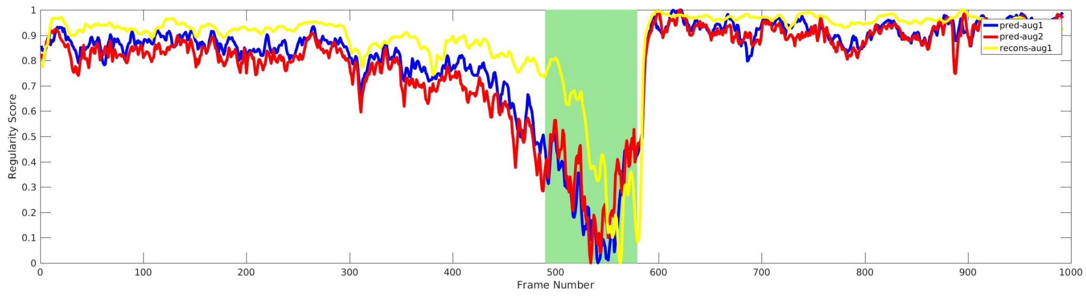
Figure 6.3: Regularity score of video sequence $\#2, 4, 5, 7$ (from top to bottom) of UCSDPed2 dataset.



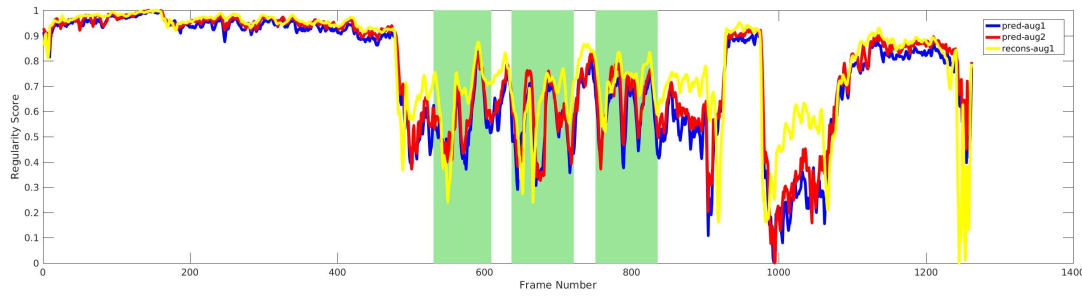
(a)



(b)



(c)



(d)

Figure 6.4: Comparison of regularity scores deriving from prediction and reconstruction errors on video sequence $\#5, 7, 15, 12$ (from top to bottom) of CUHK Avenue dataset.

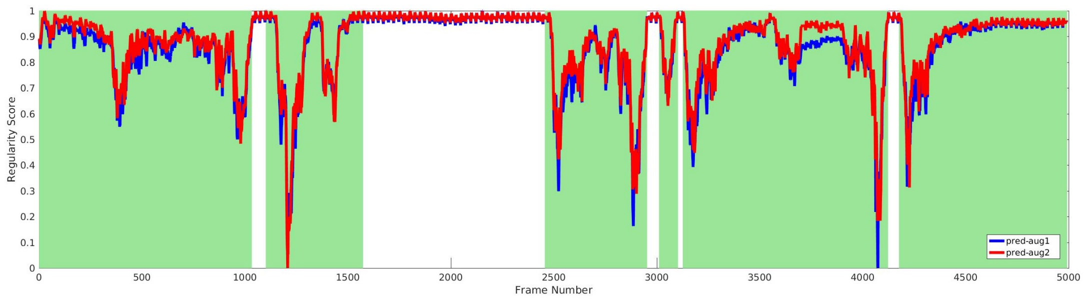


Figure 6.5: Regularity score of frames $\#115,000 - 120,000$ of Subway entrance dataset (with masking the timestamp).

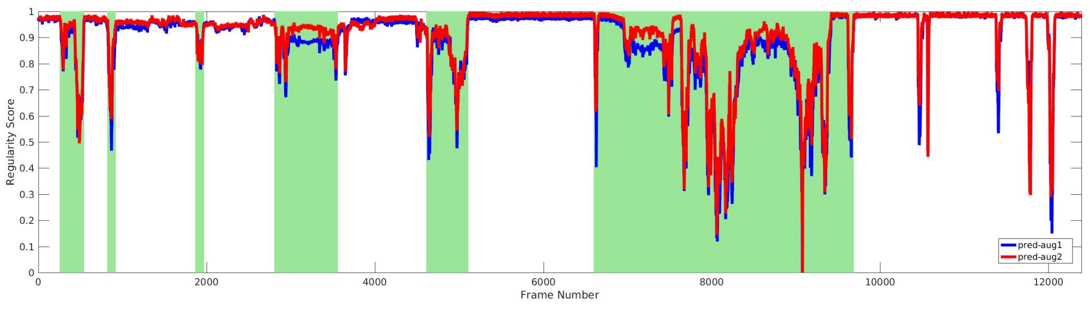


Figure 6.6: Regularity score of frames $\#52,500 - 64,000$ of Subway exit dataset (with masking the timestamp).

Bibliography

[1] Oren Boiman and Michal Irani. Detecting irregularities in images and in video. *International journal of computer vision*, 74(1):17–31, 2007.

[2] Yang Cong, Junsong Yuan, and Ji Liu. Sparse reconstruction cost for abnormal event detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3449–3456. IEEE, 2011.

[3] Hanhe Lin, Jeremiah D Deng, Brendon J Woodford, and Ahmad Shahi. Online weighted clustering for real-time abnormal event detection in video surveillance. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 536–540. ACM, 2016.

[4] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935–942. IEEE, 2009.

[5] Louis Kratz and Ko Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1446–1453. IEEE, 2009.

[6] Weixin Li, Vijay Mahadevan, and Nuno Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):18–32, 2014.

[7] David Charte, Francisco Charte, Salvador García, María J del Jesus, and Francisco Herrera. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion*, 2017.

[8] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[9] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4489–4497. IEEE, 2015.

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[11] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, et al. Fully convolutional neural network for fast anomaly detection in crowded scenes. *arXiv preprint arXiv:1609.00866*, 2016.

[12] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–742, 2016.

[13] Yong Shean Chong and Yong Haur Tay. Abnormal event detection in videos using spatiotemporal autoencoder. In *International Symposium on Neural Networks*, pages 189–196. Springer, 2017.

[14] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1933–1941. ACM, 2017.

[15] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[16] Cewu Lu, Jianping Shi, and Jiaya Jia. Abnormal event detection at 150 fps in matlab. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2720–2727, 2013.

[17] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1975–1981. IEEE, 2010.

[18] Jaechul Kim and Kristen Grauman. Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2921–2928. IEEE, 2009.

[19] Allison Del Giorno, J Andrew Bagnell, and Martial Hebert. A discriminative framework for anomaly detection in large videos. In *European Conference on Computer Vision*, pages 334–349. Springer, 2016.

[20] Miao Yu, Adel Rhuma, Syed Mohsen Naqvi, Liang Wang, and Jonathon Chambers. A posture recognition-based fall detection system for monitoring an elderly person in a smart home environment. *IEEE transactions on information technology in biomedicine*, 16(6):1274–1286, 2012.

[21] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE transactions on Intelligent transportation systems*, 1(2):108–118, 2000.

[22] Luis Patino and James Ferryman. Detecting threat behaviours. In *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th IEEE International Conference on*, pages 88–94. IEEE, 2016.

[23] Tom Cane and James Ferryman. Saliency-based detection for maritime object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–25, 2016.

[24] Teng Li, Huan Chang, Meng Wang, Bingbing Ni, Richang Hong, and Shuicheng Yan. Crowded scene analysis: A survey. *IEEE transactions on circuits and systems for video technology*, 25(3):367–386, 2015.

[25] M Sami Zitouni, Harish Bhaskar, J Dias, and Mohammed E Al-Mualla. Advances and trends in visual crowd analysis: a systematic survey and evaluation of crowd modelling techniques. *Neurocomputing*, 186:139–159, 2016.

[26] Tao Xiang and Shaogang Gong. Incremental and adaptive abnormal behaviour detection. *Computer Vision and Image Understanding*, 111(1):59–73, 2008.

[27] Bo Wang, Mao Ye, Xue Li, and Fengjuan Zhao. Abnormal crowd behavior detection using size-adapted spatio-temporal features. *International Journal of Control, Automation and Systems*, 9(5):905, 2011.

[28] Vikas Reddy, Conrad Sanderson, and Brian C Lovell. Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 55–61. IEEE, 2011.

[29] Dan Xu, Rui Song, Xinyu Wu, Nannan Li, Wei Feng, and Huihuan Qian. Video anomaly detection based on a hierarchical activity discovery within spatio-temporal contexts. *Neurocomputing*, 143:144–152, 2014.

[30] Yang Cong, Junsong Yuan, and Ji Liu. Abnormal event detection in crowded scenes using sparse representation. *Pattern Recognition*, 46(7):1851–1864, 2013.

[31] Bin Zhao, Li Fei-Fei, and Eric P Xing. Online detection of unusual events in videos via dynamic sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3313–3320. IEEE, 2011.

[32] Xiaobin Zhu, Jing Liu, Jinqiao Wang, Changsheng Li, and Hanqing Lu. Sparse representation for robust abnormality detection in crowded scenes. *Pattern Recognition*, 47(5):1791–1799, 2014.

[33] Siqi Wang, En Zhu, Jianping Yin, and Fatih Porikli. Anomaly detection in crowded scenes by sl-hof descriptor and foreground classification.

[34] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.

[35] Hanhe Lin, Jeremiah D Deng, and Brendon J Woodford. Anomaly detection in crowd scenes via online adaptive one-class support vector machines. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2434–2438. IEEE, 2015.

[36] Tian Wang, Jie Chen, Yi Zhou, and Hichem Snoussi. Online least squares one-class support vector machines-based abnormal visual event detection. *Sensors*, 13(12): 17130–17155, 2013.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical*

image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

[40] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[41] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[42] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[43] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015.

[44] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011.

[45] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016.

[46] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[47] Oluwatoyin P Popoola and Kejun Wang. Video-based abnormal human behavior recognition—a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):865–878, 2012.

[48] Angela A Sodemann, Matthew P Ross, and Brett J Borghetti. A review of anomaly detection in automated surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1257–1272, 2012.

[49] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[50] B Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *arXiv preprint arXiv:1801.03149*, 2018.

[51] Hannah M Dee and David C Hogg. Detecting inexplicable behaviour. In *BMVC*, pages 1–10, 2004.

[52] Neil Robertson, Ian Reid, and Michael Brady. Behaviour recognition and explanation for video surveillance. 2006.

[53] Mahdyar Ravanbakhsh, Enver Sangineto, Moin Nabi, and Nicu Sebe. Training adversarial discriminators for cross-channel abnormal event detection in crowds. *arXiv preprint arXiv:1706.07680*, 2017.

[54] Mehrsan Javan Roshtkhari and Martin D Levine. An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions. *Computer vision and image understanding*, 117(10):1436–1452, 2013.

[55] Xinyi Cui, Qingshan Liu, Mingchen Gao, and Dimitris N Metaxas. Abnormal detection using interaction energy potentials. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3161–3167. IEEE, 2011.

[56] Borislav Antić and Björn Ommer. Video parsing for abnormality detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2415–2422. IEEE, 2011.

[57] Borislav Antić and Björn Ommer. Spatio-temporal video parsing for abnormality detection. *arXiv preprint arXiv:1502.06235*, 2015.

[58] Neil Johnson and David Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision computing*, 14(8):609–615, 1996.

[59] Arslan Basharat, Alexei Gritai, and Mubarak Shah. Learning object motion patterns for anomaly detection and improved object detection. In *Computer Vision and*

Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[60] Fan Jiang, Junsong Yuan, Sotirios A Tsaftaris, and Aggelos K Katsaggelos. Anomalous video event detection using spatiotemporal context. *Computer Vision and Image Understanding*, 115(3):323–333, 2011.

[61] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009.

[62] Yannick Benezeth, P-M Jodoin, Venkatesh Saligrama, and Christophe Rosenberger. Abnormal events detection based on spatio-temporal co-occurences. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2458–2465. IEEE, 2009.

[63] Ying Zhang, Huchuan Lu, Lihe Zhang, and Xiang Ruan. Combining motion and appearance cues for anomaly detection. *Pattern Recognition*, 51:443–452, 2016.

[64] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[65] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[66] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[67] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

[68] Robert H Shumway and David S Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.

[69] Peter Van Overschee and Bart De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1): 75–93, 1994.

[70] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

[71] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[72] Michael E Tipping and Christopher M Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[73] Antoni B Chan and Nuno Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):909–926, 2008.

[74] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2007.

[75] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[76] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[77] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[78] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11-13):1191–1199, 1999.

[79] Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[80] Young-Sik Choi. Least squares one-class support vector machine. *Pattern Recognition Letters*, 30(13):1236–1240, 2009.

[81] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[82] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[84] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[85] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.

[86] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[88] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[89] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 2011.

[90] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[91] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[92] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[93] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[94] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[95] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[96] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[97] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[98] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[99] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014.

[100] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[101] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE, 2015.

[102] Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017.

[103] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[104] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[105] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[106] Jefferson Ryan Medel. *Anomaly Detection Using Predictive Convolutional Long Short-Term Memory Units*. Rochester Institute of Technology, 2016.

[107] Amit Adam, Ehud Rivlin, Ilan Shimshoni, and Daviv Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE transactions on pattern analysis and machine intelligence*, 30(3):555–560, 2008.

[108] Alireza Makhzani and Brendan J Frey. Winner-take-all autoencoders. In *Advances in Neural Information Processing Systems*, pages 2791–2799, 2015.

[109] Ce Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Citeseer, 2009.

[110] Matconvnet: Cnns for matlab. `http://www.vlfeat.org/matconvnet/`.

[111] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[112] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. *arXiv preprint arXiv:1706.03458*, 2017.

[113] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems*, pages 2802–2810, 2016.

[114] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[115] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[116] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[117] Hanh TM Tran and DC Hogg. Anomaly detection using a convolutional winner-take-all autoencoder. In *Proceedings of the British Machine Vision Conference 2017*. Leeds, 2017.

[118] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[119] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.