Anomaly Detection from Videos : A Deep Learning Approach

Seby Jacob

Master of Engineering

Department of Electrical and Computer Engineering

McGill University

Montreal,Quebec

August 15, 2018

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Engineering

DEDICATION

This document is dedicated to my parents, Jacob and Mary, for their inexhaustible enthusiasm in enabling my pursuit of happiness.

PREFACE

This thesis presents original scholarship by the author. The results, analysis and views reported throughout, reflect work done largely by the author with assistance from Prof.Martin D. Levine at McGill University. The algorithms presented were implemented by the author except where explicitly denoted otherwise.

ACKNOWLEDGEMENTS

Work towards this thesis began in September 2016 and continued until May 2018 with minor revisions performed until August 2018. All through this time, I have had to face a number of obstacles and challenges. The support from my friends and family has been invaluable in helping me conquer them all. First, I would like to thank my parents, Jacob Raj and Dr Mary Dorothy, for their love and support, without which I would not have been able to pursue my graduate studies at McGill University. Second, my advisor Prof. Martin D. Levine has been indispensable to me, having given me the confidence to pursue this work as well as the support and guidance I needed. Whenever I encounter an issue or a roadblock, he is always available for consultation and advice, while remaining open to new ideas and research directions. He regularly goes beyond what needs to be done to send me interesting literature that would further this work. I especially appreciate his ability to summarize complex problems into fundamental mathematical ideas to complement my algorithmic thinking. I am extremely appreciative to Jan Binder and Nick Wilson for their assistance with any computer related issues and for willing to accommodate some of my esoteric requests. My colleague Michael Smith was of exceptional help to me during the period of this work. He frequently shared his computational resources for running my experiments when there were serious resource crunches. Relevant gratitude needs to be expressed to my colleagues Nazanin Mohammadi and Dimitrios Gallos, who willingly shared their expertise in theory and programming to help my research. Last, but not least, I would like to thank all the McGill staff that helped make this journey of mine devoid of any hitches.

ABSTRACT

This thesis proposes an innovative solution to detect and localize anomalous events in a video stream from a static camera. Anomalies are defined as events with a very low probability of occurrence in the scene or as events typically uncharacteristic of the scene. In this work, we employ a constrained convolutional auto-encoder to learn the scene characteristics. The autoencoder is trained on spatio-temporal video-volumes extracted from recorded videos of the scene. Once the training is complete, each incoming video-volume can be tested for its anomalous nature by analyzing the low-dimensional encodings and the quality of its reconstruction from the auto-encoder.

Anomalies are heavily subjective to the scene being monitored. The most abnormal event in one scene could be the most normal event in another. Hence, special care has been taken to make the solution applicable for any scenario. Since training is unsupervised, this work is extremely general purpose and can be deployed on any scene as is. Apart from the discourse on a novel solution that is competitive with state-of-the-art methods, this work also has an additional contribution. Specifically, we present a framework for generating unlimited amounts of video data for anomaly detection from a static camera. This enables the evaluation of any deep learning models, that were previously not adaptable for the problem due to the limited training data available in benchmark datasets.

We present results from extensive experimentation on popular benchmark datasets to show that our solution is effective and robust for anomaly detection. We also establish the importance of having sufficient training data via the evaluation of models trained on training-sets of varying sizes. Finally, the idiosyncratic nature of "What is an anomaly?" is subjected to analysis using an experimental methodology.

ABRÉGÉ

Cette thèse propose une solution innovante pour détecter et localiser les événements anormaux dans un flux vidéo provenant d'une caméra statique. Les anomalies sont définies comme des événements avec une très faible probabilité d'occurrence dans la scène ou comme des événements atypiques . Dans cette étude, nous utilisons un auto-encodeur convolutionnel limité pour analyser les caractéristiques de la scène. Le codeur automatique repose sur des volumes vidéo spatio-temporels extraits de vidéos enregistrées de la scène. Une fois l'entraînement terminé, chaque anomalie du volume vidéo entrant peut être testée en analysant les codages à faible dimension et la qualité de sa reconstruction à partir de l'encodeur automatique.

Les anomalies sont fortement subjectives pour la scène surveillée. L'événement le plus anormal d'une scène pourrait être considéré comme étant le plus normal dans une autre. Par conséquent, une attention particulière a été apportée pour rendre la solution applicable à tous les scénarios. Puisque la formation n'est pas supervisée, cette recherche est extrêmement générale et peut être déployée sur n'importe quelle scène. Hormis le travail sur une nouvelle solution compétitive par rapport aux autres méthodes, ce dernier apporte également une contribution supplémentaire. Nous présentons un cadre pour générer des quantités illimitées de données vidéo afin de détecter des anomalies à partir d'une caméra statique. Cela permet d'évaluer de manière exhaustive tous les modèles d'apprentissage, qui auparavant n'étaient pas adaptées au problème en raison des données de formation limitées disponibles dans les ensembles de données de référence.

Nous présentons les résultats d'une expérimentation basée sur des ensembles de données de référence populaires pour montrer que notre solution est efficace et solide en vue de détecter des anomalies. Par ailleurs, nous établissons l'importance de disposer de données de formation suffisantes à travers l'évaluation de modèles construits sur des ensembles de formation de différentes tailles. En définitive, la nature idiosyncratique de "Qu'est-ce qu'une anomalie?" est soumise à une analyse en utilisant une méthodologie expérimentale.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Background

Modern society is swarmed by smart-phones and innumerable cameras. The amount of video data being generated day to day has reached tremendous heights. For instance, 300 hours of video data are uploaded to YouTube every minute [5]. The high volume of video data generates a proportional demand for top quality video-processing algorithms. Some practical examples of video-processing applications include video surveillance systems, person identification and tracking, action recognition, video annotation, and video summarization. This thesis proposes a unique solution for implementing a general purpose, automatic, video surveillance system.

The drop in prices for security cameras and hard-disk drives, coupled with increasing security concerns has fueled a boom in the security and surveillance industry. The global video surveillance market has grown from $18 billion in 2009 to $49 billion in 2018 [67]. Organizations, institutions, stores and even a large number of homes in developed cities make use of closed circuit cameras to oversee their premises. However, most of these cameras are not subject to active live monitoring. Even if they are, active live monitoring is a huge waste of human effort, as noteworthy events seldom happen. Usually, the recorded data are used only as hindsight after a crime or an unfortunate event has already occurred. This realization has induced a need for developing reliable, real-time, video surveillance algorithms. These algorithms must forward only the events of interest to a human arbiter, who can then take immediate steps when necessary. This thesis details an effort in developing a video surveillance solution that encapsulates these pertinent features.

1.2 Problem statement

The problem under consideration may be effectively summarized as: Anomaly detection in surveillance videos. Surveillance videos are usually recorded using a static camera overlooking a fixed scene where events occur. Events in a surveillance video can be categorized into:

- Normal events

  These are events characteristic to the scene. These events are not worthy of intervention or any action from humans. Consequently, these are the most frequent events, or the only type of events that occur in the scene.

- Abnormal or anomalous events

  These events are uncharacteristic for the scene. They are unusual, irregular or suspicious activities. Consequently, the occurrence of such an event requires human intervention. These events do not occur frequently (or they never occur), and are hence referred to as anomalous events.

Consider a static camera looking at an ATM cubicle. Normal events in this scene would include people withdrawing money and doing bank transactions at the terminal. On the other hand, anomalous events could include a gun being pulled on a customer, someone destroying the terminal with a hammer, or a small group of people trying to play mini-golf in the cubicle.

Given an ATM cubicle, someone pulling a gun or destroying the terminal are events *more probable* than some people engaging in a casual game of mini-golf. Does this mean that the mini-golf event is much more important to be detected than a gun being pulled on someone? How can we assign a scale of abnormality to each event? Should it be based on how urgently a situation demands human intervention, or how improbable the event actually is? These questions raise an important issue: The subjective nature of the scale of abnormality.

Even though the scale of abnormality is subjective, all the examples of abnormal events given above are irregular occurrences in an ATM cubicle. Therefore, they are all to be classified as anomalous events by a video surveillance system.

1.2.1 Anomaly detection

In order to engineer a solution to detect anomalous events, we require a mathematical notion for an anomalous event. *Following the discussion above, an anomalous event can be implicitly defined as an event with an extremely low probability of occurrence, when compared to previous events observed in the scene.* This definition stems from the assumption that events which occur very rarely in a scene are potential threats that need attention. Detection of an anomalous event includes two goals, namely, identifying an event as anomalous and localizing the manifestation of the identified event within the monitored space.

Video surveillance can be employed anywhere a video camera can be installed. Hence, a reliable video surveillance algorithm must be applicable to any scene. The algorithm must perform well on any premises being monitored. Therefore, as an additional goal for this work, we aspire to make our system unbiased to any scene it is monitoring.

1.2.2 Deep learning

When choosing a foundation to build the solution, it was an easy choice between deep learning and classical video processing algorithms. Classical video processing algorithms are limited to using general purpose hand-crafted feature extractors to learn the nature of the scene. These feature extractors may not be able to successfully capture the behaviour of certain scenes, as they require heavy fine-tuning to be adapted for each scene. Deep learning may help solve this problem by extracting the relevant features that can best understand the scene. Currently, deep learning is successfully utilized in high data volume computer-vision tasks like image classification, object recognition, face recognition, image captioning, image segmentation, video annotation, video captioning, etc. However, there are hardly any *reliable* anomaly detection algorithms that employ deep learning. We chose deep learning for this work, as this would further the field of deep learning and computer-vision research, while harnessing its proven power in high data volume applications.

1.2.3 The detailed problem statement

Summarizing all the ideas in 1.2, the detailed problem statement is as follows: *Automatic detection and localization of events with very low likelihood of occurring in surveillance videos from a static camera, using deep learning.*

1.3 Terminology

- Anomaly: Something that deviates from what is standard, normal, or expected.
- Autoencoder: A model that is trained to reconstruct the input.
- Convolution: Convolution can generally be defined as an integral that quantifies the amount of overlap of one function g as it is shifted over another function f. For the purposes of this thesis, we restrict convolution to its definition in the computer-vision domain. In computer-vision, convolution is a weighted multiplication of the image and a kernel matrix to generate a feature-map.
- Convolutional autoencoder: A deep learning model that uses convolutions to extract features from an image, and uses the extracted features to reconstruct the image.
- Cuboid: In this thesis, a cuboid represents a 3-dimensional array.
- Spatio-temporal: Of both space and time.
- Spatio-temporal cuboid: A cuboid that contains data specific to a point in space at a given time.

Chapter 2
Related work

For a complete overview of anomaly detection, the reader is directed to the work of Chandola *et al.* [10]. It presents a structural and comprehensive overview of the research on anomaly detection in all types of data. They study the main algorithms used in every field, while analyzing the nature of anomalies in each. Every method is deconstructed and the key assumptions and conditions for its performance are examined.

In this chapter, we will restrict ourselves to examining the area of anomaly detection *from videos*. We will highlight the main methods of video representation, problems, and techniques that have been addressed in the field in two separate high-level branches, namely the classical computer vision approach and the deep learning approach.

2.1 Modeling representations of videos

It is nearly impossible to process high-dimensional video data using a classifier due to excessive computational complexity. Raw video data usually has a lot of redundancy in its content. These facts demonstrate the need for representing video data in a low-dimensional manifold to be processed fast and effectively. The low-dimensional representation must also retain enough information to perform the given task well. The performance of any application is usually related to the quality of the representation. However, selecting a representation for a task can be heavily challenging due to the high degree of freedom in the 3-D space from where video is captured. The existence of environmental variations and noise is also counter-productive in helping us choose a good representation of the video data.

Before the emergence of deep learning as a popular method for feature extraction, pre-defined (or hand-crafted) features were used to represent the video data in a relatively low-dimensional manifold for anomaly detection. Popular methods like histograms of motion, direction, and optical flow have been used for object detection and tracking in videos to enable the detection of irregular behaviours.

However, these features need to be tuned each time a different scene is studied. In other words, the pre-defined features might not be the best in capturing the pertinent motion information given *any scenario*. This makes deep learning methods for feature extraction a very good candidate, owing to its capability of producing robust features for any given data input.

In the following sections, we will look at two different classes of anomaly detection algorithms. First using classical computer vision, followed by deep learning methods.

2.2 Classical features for video representation

The word classical in the title is not to be confused with out-dated. Handcrafted features are used widely in computer vision applications owing to their high efficiency, while consistently providing low-computation algorithms. This section will review methods in the literature that use classical vision algorithms for anomaly detection.

2.2.1 Object trajectory

Considering the assumption that anomalies are associated with motion, one of the main methods used in anomaly detection is the study of object trajectories in the scene. The authors of [30] detect and track objects whose trajectories are modeled by Hidden Markov Models (HMMs). These time-series models are then subject to unsupervised clustering based on the cross-likelihood-ratio [31]. A trajectory is classified as unusual based on the probability of the trajectory, given the normal distribution of trajectories. However, this method breaks down when clearly visible trajectories are unavailable or if the scene is crowded.

A similar framework is employed by [74] , which aims to detect deviant walking paths in a shopping mall corridor. Person trajectories are modelled as a set of points in space and then clustered using their similarities, measured by the Hausdorff distance. The walking paths from clusters with very low memberships are then determined to be anomalies.

Adding a slight twist to the idea of using trajectories themselves, [18] describes a dynamic oriented graph classifier that detects important points in the trajectories and identifies them as nodes. Each trajectory is then modelled as a graph in this space of nodes. Infrequent graphs are then classified as anomalies. [44] describes an automatic real-time model for anomaly detection using the altruistic vector quantization algorithm (AVQ). The proposed algorithm is capable of encoding the trajectories of moving blobs in any scene into a prototype. The number of prototypes are determined autonomously. The learned prototypes encapsulate the visual behaviour of dynamic objects in the scene. If any query prototype significantly deviates from the learned prototypes, an alarm is raised.

2.2.2 Histograms of differential measures from images

One of the most popular works in anomaly detection using spatial histograms of motion information is described in [85]. The authors employ spatio-temporal motion filtering of video to compute the histograms. These histograms are classified into prototypes, from which a prototype-segment co-occurrence matrix is computed. The inferred similarity of features from test samples are calculated in the co-embedding space and isolated clusters are labelled unusual.

The authors of [35] provide a framework for a video analysis tool for suspicious event detection. As a first step, they suggest human labelling of suspicious activity in previous video events. From the events labelled as aberrant, spatio-temporal intensity histograms are learned. Any future event that creates similar spatio-temporal intensity histograms is then classified as an anomaly. Unlike the latter, [82] proposes the use of semi-supervised, adapted, HMMs for unusual event detection. This method involves learning a K-class HMM and requires ground truth test labels supplied by a human. The HMM is trained to maximize the likelihood of the motion histograms from the normal events. The HMM adapts its states based on the information supplied by a human and is then later used to detect unusual activity.

The monitoring of human activity has the parallel objective to detect and take action when unusual activities take place. An instance of this timely intervention is coming to the aid of a resident who has fallen down in an old peoples home. [21] describes such an application, dedicated to detecting the fall of patients in such a facility. First, the segmentation of moving objects is obtained by background elimination. This leaves only motion silhouettes which are then studied for shape changes. An ellipsoid is approximated around each moving object. A combination of the approximated ellipse and the horizontal and vertical projection histograms of the motion silhouette are used to train a supervised neural network to accurately classify each type of motion. When an action is classified as a fall, the authorities are alerted.

2.2.3 Spatio-temporal features

Inspired from the use of HMMs to model a time-series, [33] uses separate HMMs to model different spatial regions in the frame. The motion variation of each local space-time volume, coupled to its spatio-temporal statistical behaviour, is modelled to characterize the behaviour of the scene. Anomalous events are detected as statistical deviations from the norm.

The authors of [7] analyze events and behaviours at the pixel level after background subtraction. In this work, events are considered to be random-variables influencing spatio-temporal statistics. Based on the motion data at the pixel level, co-occurrence statistics are learned across space-time for the scene. This is then used as a potential function in a Markov Random Field (MRF) to describe the conditional probability of observations in each spatio-temporal volume. The posterior probability for a new sample is then used for classification.

Roshtkari and Levine, [55] define anomalies as spatio-temporal compositions having low probability of occurrence with respect to the previous observations. In their work, they sample spatio-temporal cuboids from the video at multiple scales and formulate the problem as a bag-of-video-words by constructing a codebook and using a weighted codeword assignment for each spatio-temporal volume. The prior probability of each codeword is calculated using the frequency of codewords seen already. These statistics are used to predict the probability of a spatio-temporal volume to be normal. Albeit simple, this method is fast and demonstrates good performance on benchmark datasets.

On the other hand, Zhao and Li [83] extract spatio-temporal interest points from a temporally sliding window in the video to learn a basis dictionary. This dictionary is learned from the video using sparse coding and is updated in an online fashion when more data are available. Each event query returns a reconstruction weight vector from the basis dictionary which can be used to construct a normality measure. Since the dictionary can be learned real-time and online, the algorithm needs to pass through the video only once. This also makes sure that concept drift in the scene is taken care of (for example, a change of season).

2.2.4 Optical flow

The final class of major classical feature extraction from videos includes the use of optical flow as a means of encoding motion statistics. A method for automatically detecting unusual human events on stairs from video data is presented in [65]. They compute two sets of features from a video of a person descending a stairwell. The first set includes foot positions and velocities. Both feet are tracked using a state particle filter based on histogram of oriented gradients. The second set of features are parameters of mean optical flow in the foreground. These features are then used to train a HMM on normal stair use, and a threshold on sequence likelihoods is used to detect unusual events in new data.

Similarly, [75] introduces a method to learn the velocity statistics of each pixel using optical flow, which is later used to identify pixels with very fast motion not encountered before. This method assumes high velocity pixels as representing abnormal behaviours. On the other hand, [32] proposes a space-time MRF to detect abnormal activities in the video. The normal patterns of activity at each local node are captured by the distribution of optical flow that is used to train a mixture of probabilistic principal component analyzers.

For all incoming test samples, the learned model is used to compute a maximum a posteriori estimate to evaluate a normality score at each local node.

Some other classical features seldom used in the domain include eigen-motion [20], visual words, [69] and 2D contour angles [37].

2.3 Deep learning for video representation

The second class of techniques, used to extract features from videos, employ the use of deep learning and neural networks. In recent years, feature extraction in computer vision tasks have taken a favourable turn towards the use of deep learning, rather than classical features. The availability of large computation banks and GPU arrays have assisted this transition.

2.3.1 Neural network autoencoders

Sabokrou *et al.* has published a series of works in using autoencoders as a means of feature extraction from videos for anomaly detection. In [59] and [60], they suggest the use of a cascade classifier. A primitive first stage classifier, trained on the similarity statistics between spatio-temporal cuboids using Structural Similarity (SSIM), is used to eliminate 75-80% of test samples very quickly. Cuboids that do not pass this test are then encoded by a sparse encoder (from a trained autoencoder) and tested for being outliers.

Combining the best of both classical and deep learning features, [25] uses a combination of histograms of oriented gradients and histograms of optical flow from the raw video frames to train a deep autoencoder. Instead of using the encodings, when a new test sample is to be tested, the reconstruction error is used to classify frames with very high pixel error rate as frames comprising an anomalous event. This proposed method can only detect anomalous frames and fails to localize the anomaly in the spatial extent of the frame.

2.3.2 Convolutional networks and autoencoders

Adding to their previous work, Sabokrou *et al.* uses a convolutional network, pre-trained for the Imagenet classification task, to extract features from video-frames in [61]. The extracted features and spatial feature maps at intermediate layers are then used to fit two

Gaussian classifiers for anomaly detection. Similarly, [77] uses a convolutional autoencoder to reduce the dimensions of image frames and dynamic images (motion image) to encode the appearance and motion features. These features are fused to train a One-Class Support Vector Machine (SVM), which is then used to classify the query samples as normal or abnormal.

The authors of [25] also present a convolutional autoencoder trained on temporal video cuboid volumes. However, in contrast to the schemes above, they use the reconstruction error on the test samples as a metric to enable the classification of unusual frames. These works prove the use of 2-dimensional convolutional autoencoders as a reliable way for feature extraction to represent videos.

Adding another dimension to the convolution kernels, the authors of [84] use 3-dimensional convolution kernels to incorporate the temporal dimension as well. The encoded features are used to train two decoders. One decoder is trained to predict future frames and another decoder to reconstruct the input. The parallel training of these two branches makes sure that the encoded features have a temporal understanding of the scene (as it should be able to predict the future frames as well as reconstruct the input frames). Just as above, they use the reconstruction error as a metric to be thresholded for anomaly assertions.

2.3.3 Convolutional long short term memory (LSTM) networks

Convolutional LSTM networks use the LSTM formulation to model the recurrent features of the input. [13] demonstrates such a technique, later using the reconstruction error to decide the normality score of the query samples. They show good performance metrics on benchmark datasets, but convolutional LSTM networks are costly to train and need extremely large computational power to be trained and deployed.

Similarly, [45] uses a Convolutional LSTM network to predict the future frames. The similarity between the predicted frames and the actual frames that arrive in the future is used to evaluate a normality score for each pixel in the spatial extent of the frame. However, this proposal also comes at an exorbitant computational cost.

2.3.4 Generative models

There has also been some exploration in training generative models to capture the underlying distribution of video data. The intuition is that if the generative model can generate samples from the training distribution, it is inevitable that the model will learn the distribution itself. This is then leveraged to assert if an incoming sample belongs to the usual or unusual events. [4] employs a trained Variational Autoencoder (VAE) to infer the reconstruction probability of a test query while [62] and [17] train Generative Adversarial Networks (GANs) and use its discriminator as a classifier to identify the anomalous samples. However, none of these examples show any advantage of using a generative model, compared to other methods.

In this thesis, we will use a deep 2-D convolutional autoencoder to represent the video data in a low-dimensional manifold.

Chapter 3
Datasets

In this chapter, we will discuss the datasets used to evaluate our solution. We have only used datasets with ground truth annotation of the anomalous events. The discussion will be tackled in two parts. First, we will explore the benchmark datasets, followed by a deliberation on the creation of an arbitrarily large dataset for anomaly detection.

3.1 Benchmark datasets

The era of deep learning has ushered in a need for high-volume datasets in computer vision. This demand has been met sufficiently in most vision research areas. The number of training observations in a high-volume dataset is usually comparable to the number of parameters in a deep neural network. For instance, the Imagenet dataset [16] for image-classification has about 1,034,908 annotated natural images over 1000 object classes. The Street View House Numbers (SVHN) dataset [49] consists of 600,000 digit images for digit classification in natural scenes. However, there are no such high-volume datasets for the video surveillance task, making it quite difficult to employ deep learning toward this goal. The benchmark datasets we use to evaluate our solution follow below.

3.1.1 University of California San Diego (UCSD) pedestrian dataset

The UCSD pedestrian dataset [9] contains videos overlooking walkways. The dataset includes videos captured from stationary cameras at two locations. The data are available as 8-bit, greyscale video frames. The normal events include pedestrians on walkways. Anomalies in the dataset consists of non-pedestrian entities and abnormal pedestrian patterns. Some examples of non-pedestrian entities are bikers, skaters, people on wheelchairs and small carts. People walking on the grass are also anomalies.

These videos are recorded in two different scenes, dividing the dataset accordingly into two.

- UCSD Ped 1

  The videos from the first scene are collected into a subset commonly referred in literature as UCSD Ped 1. This includes video clips of groups of people walking towards and away from the camera. This section contains 34 training video samples and 36 testing video samples. Some examples are depicted in Fig 3–1. There are about 6800 training frames and 7200 testing frames available from this scene.

- UCSD Ped 2

  The videos from the second scene are commonly referred to as UCSD Ped 2. This includes video clips of people walking parallel to the camera plane. This section contains 16 training video samples and 12 testing video samples. Illustrations of this dataset can be seen in Fig 3–2. This subset consists of 3200 training frames and 2400 testing frames.

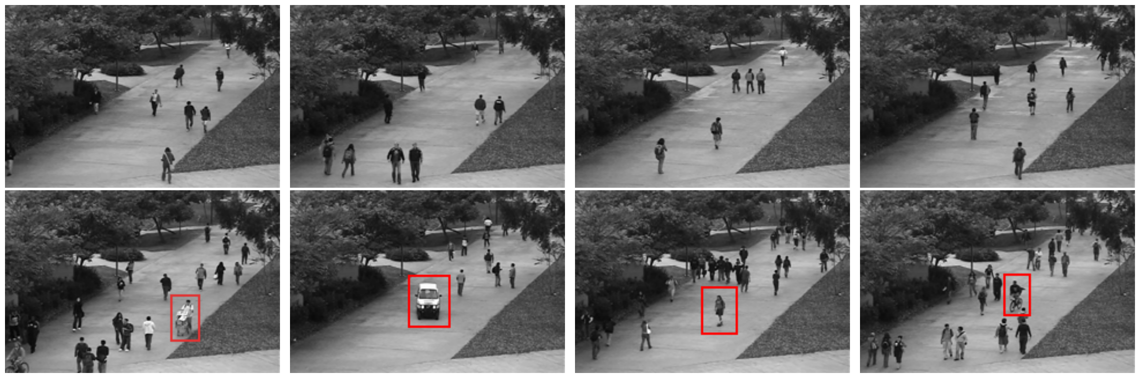


Figure 3–1 – Examples of images from the UCSD Ped 1 collection. Top Row: Images from the training set without any anomalies. Bottom Row: Images from the test set with observed anomalies marked in red. From left to right, the anomalies are a trolley, a van, skateboarding and bicycling.

3.1.2 York anomalous behavior dataset

The York anomalous behaviour dataset [80] consists of videos from 8 different scenes. They are all recorded from static cameras with challenging scenarios like varying illumination effects, scene clutter, variable target appearance, rapid motion and camera jitter. Out of the available 8 image sequences eliminate 3 sequences due to the lack of clearly defined anomalous behaviours in them.

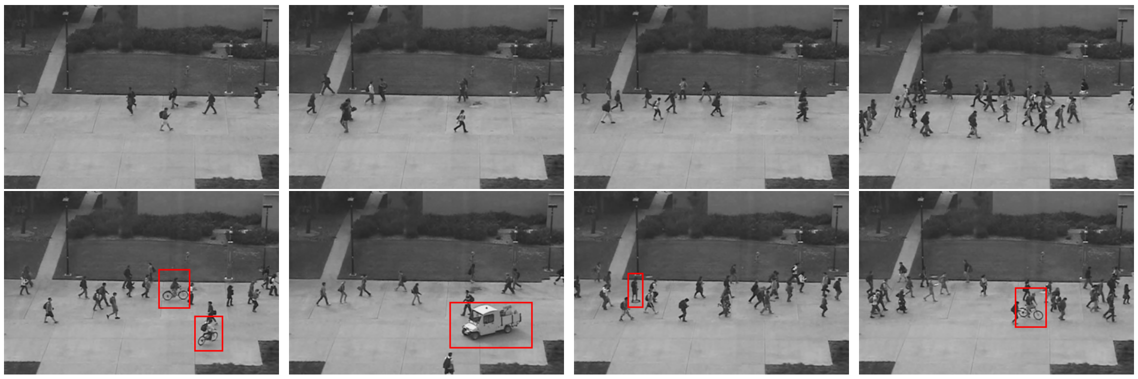


Figure 3–2 – Examples of images from the UCSD Ped 2 collection. Top Row: Images from the training set without any anomalies. Bottom Row: Images from the test set with observed anomalies marked in red. Left to right, the anomalies are bicycling, van, skateboarding and bicycling.

It is key to note that the events to be detected as anomalies, according to this dataset, are actually normal events when considering the nature of the scene. However, they are classified to be anomalous due to the very low frequency with which they occur in the footage. In other words, the anomalous events in the dataset are subjective to each scene.

- Boat-Sea

  This video contains 450 frames from a camera overlooking a bridge in the sea. A boat floating into the scene is the only anomalous event in this sequence. Fig 3–3 contains examples of frames from the dataset.

- Train

  This is an 18.5k frame sequence from a surveillance camera on a passenger train. Normal events include people sitting in the seats, whereas infrequent occurrences include movement in the aisles. Fig 3–4 captures some illustrations from this dataset.

- Boat-River

  A very short sequence from a river with a boat coming across the camera plane as the anomaly to be detected. Refer Fig 3–5 for examples.

- Canoe

  Very similar to the Boat-River dataset. A short sequence recorded over a river with a canoe coming across the field of view as the anomaly. Illustrated in Fig 3–6.

- Camouflage

  An interesting video where a person wearing camouflage that matches the background is going to and fro across the frame. The task is to detect the person when he is in the frame. Fig 3–7 has some instances from the data.

Dataset	Color scheme	Total training frames	Total testing frames
UCSD Ped 1	Greyscale	6800	7200
UCSD Ped 2	Greyscale	3200	2400
Boat-Sea	Color	200	250
Train	Color	500	18000
Boat-River	Color	100	235
Canoe	Color	200	800
Camouflage	Color	20	1440

Table 3–1 – Size and nature of benchmark datasets.

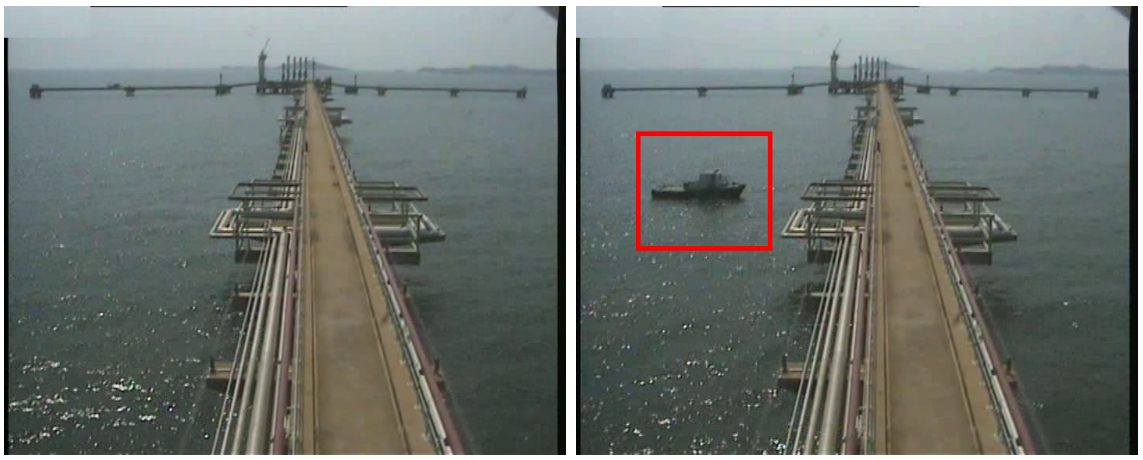


Figure 3–3 – Examples of images in the Boat-Sea dataset. On the left, a normal image without any events. On the right, anomalous event of a boat to be detected.

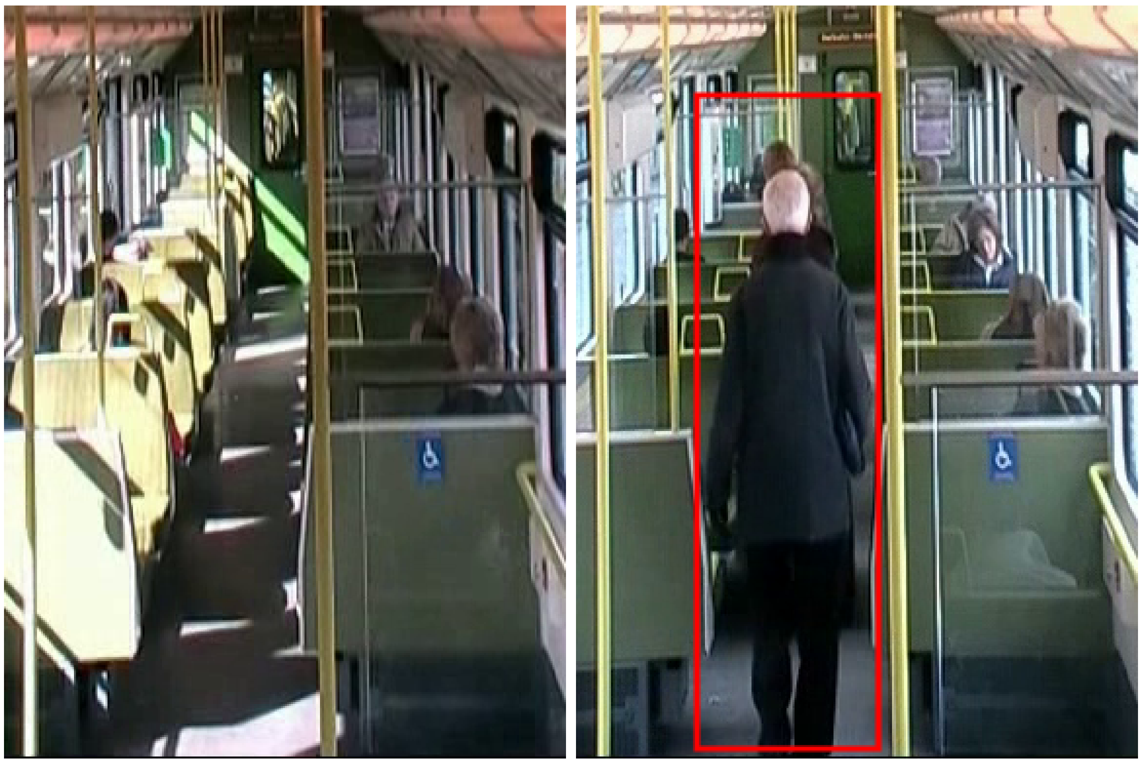


Figure 3–4 – Examples of images in the Train dataset. Left: a normal image without any anomalies. Right: anomalous event of movement in the aisles.

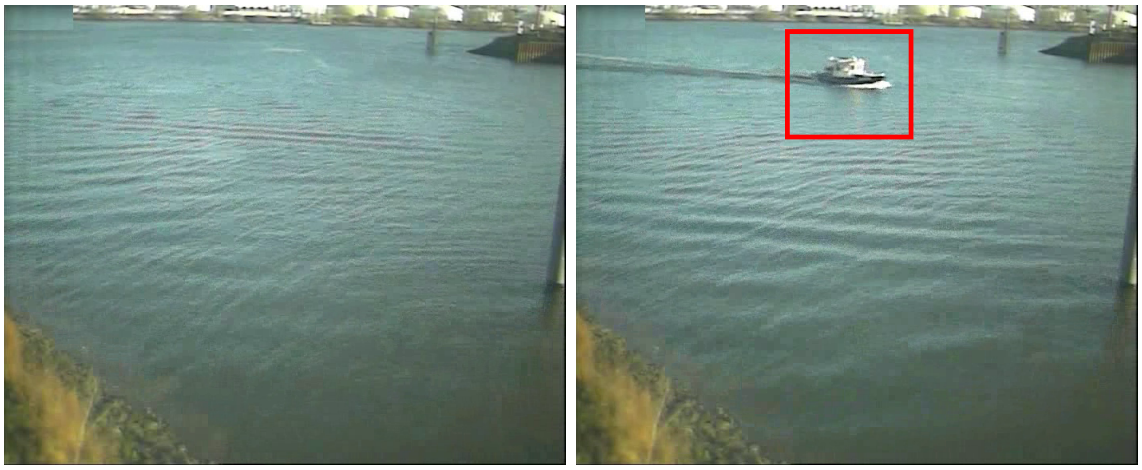


Figure 3–5 – Examples of images in the Boat-River dataset. On the left, a normal image without any events. On the right, anomalous event of a boat to be detected.

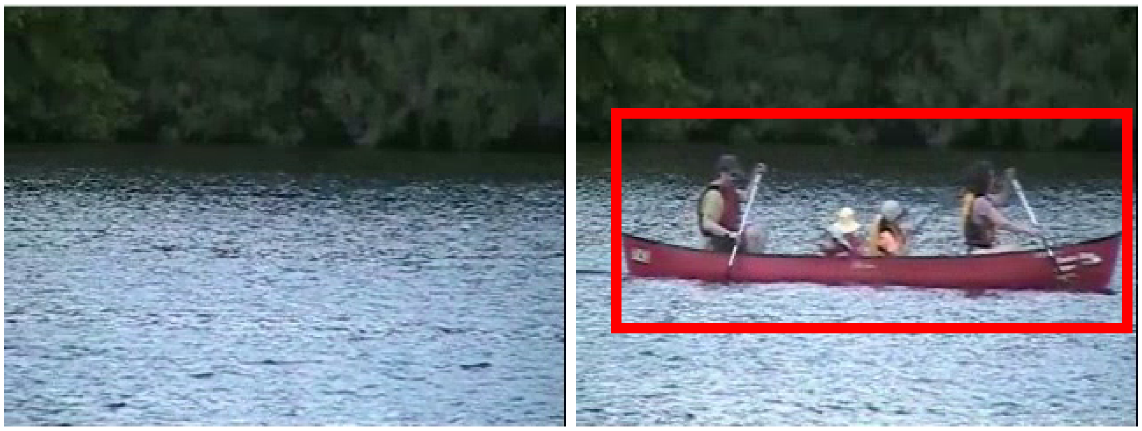


Figure 3–6 – Examples of images in the Canoe dataset. Left: a normal image without any anomalies. Right: anomalous event of a canoe moving across the field of view.

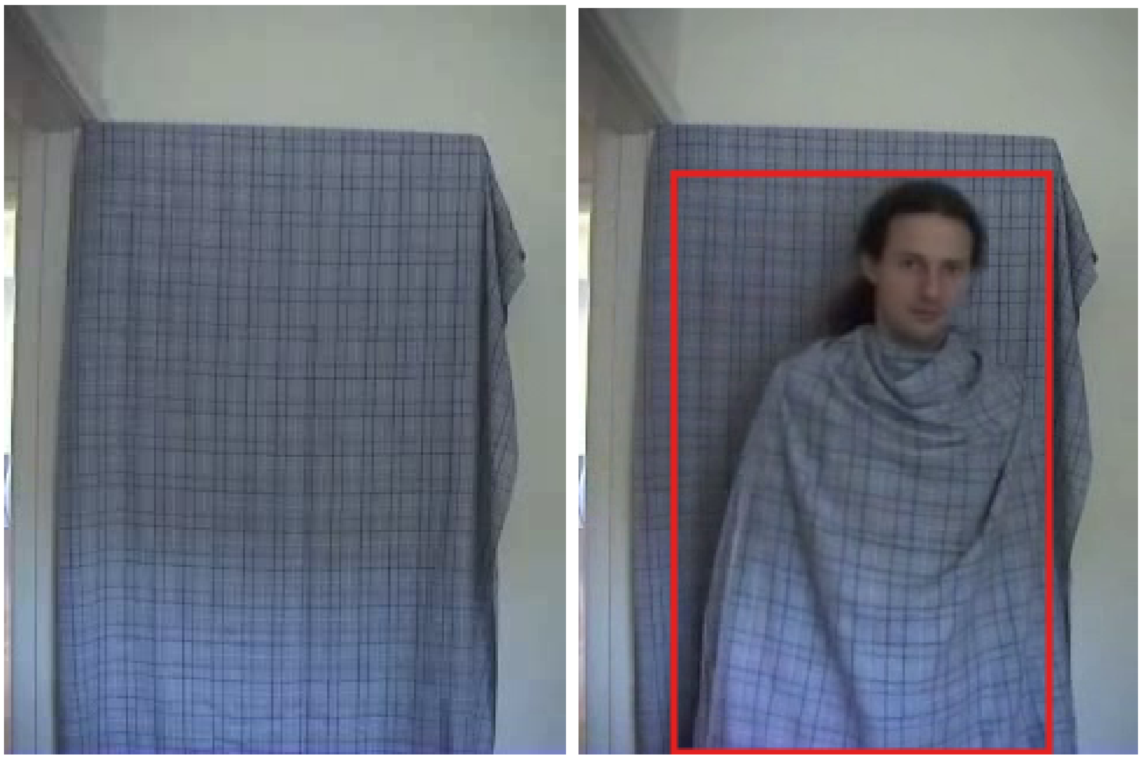


Figure 3–7 – Examples of images in the Camouflage dataset. Left: a normal image without any anomalies. Right: anomalous event of a person moving across the field of view.

3.2 Artificial data for deep learning

To understand the need for artificial data, we need to compare the benchmark datasets in the video surveillance area to benchmarks in other computer-vision areas. Let us take a brief look at the image classification domain. Imagenet Large Scale Visual Recognition Challenge (ILSVRC) [58] is a very popular benchmark for image classification. ILSVRC has been held every year since 2010 by the Imagenet project. Multiple software projects compete in the challenge to correctly classify and detect objects in images.

AlexNet [34], the first successful attempt using deep learning to solve the challenge, was a major breakthrough in the competition, to the extent that it is considered to be the beginning of the recent deep learning revolution. The Imagenet dataset training split used in the challenge consists of 1.3 million high-resolution images. Some popular, deep convolutional networks that have held first place in the challenge over the years, can be studied from Table 3–2. From this observation, it is evident that the availability of a high-volume, top-quality dataset enables the use of complex models with a large number of parameters. Unfortunately, no such high-volume dataset exists in the field of video surveillance. Even if a full-length surveillance footage is captured, the number of notable events might be very low. There is also a lot of effort involved in annotating and generating the pixel level ground truth of anomalies for evaluation. In this work, we ameliorate this situation by creating a program for generating a high-resolution, high-volume, artificial dataset with ground-truth annotations.

CNN (Model)	**Parameters in the convolutional layers**
VGG-16 [64]	14 million
VGG-19 [64]	20 million
Xception-V1 [12]	21 million
ResNet50 [26]	23 million
InceptionV3 [68]	22 million

Table 3–2 – Deep CNNs that won the ILSVRC challenge.

3.2.1 Artificial data generation

Data from every surveillance camera has two components, namely the static component and the dynamic component. The static component comprises the background of the scene and the dynamic component encompasses all the motion in the scene. The dynamic component includes moving objects and motion in the static components (for example: leaves blowing in the wind). The artificial data we generate is founded on these basic assumptions.

The data are generated using a program written in Python [56]. The program initializes a background simulating a surveillance camera over-looking a road as in Fig 3–8. This is the static component of the scene. The dynamic nature of the scene is composed of randomly colored shapes (circles and squares) moving along the two lanes of the road in queues. The inter-shape distances in each queue is random. This is done with the intention of simulating traffic on a busy road.

The program generates frames sequentially, moving the shapes one pixel at a time. This simulates a high sampling rate from the imaginary camera. The length of the training dataset can be determined by the user. This provides great flexibility and a source for high-volume training data. Consequently, it enables the training of complex deep learning models.

- Training data

  The training data consists of randomly colored squares and circles moving in two queues along the road. The squares are moving left to right in the top half of the image and the circles are moving right to left in the bottom half of the image. Some images from the training dataset can be observed in Fig 3–9.

- Test data

  The test data mostly consists of randomly colored squares and circles moving along the road. The queues are interspersed with randomly colored triangles with a very low probability that can be set by the user. The presence of the triangles are the anomalies that are to be detected since the training data contains only squares and circles. Since the anomalies are generated programmatically, the pixel level ground truth for the anomalies can be generated as bitmaps. Some images from the test dataset can be observed in Fig 3–10. As seen, the pixels occupied by triangles in the frames are marked out as anomalies in the ground truth annotations.

It is imperative to note that the colours of the shapes are of no consequence to us. All shapes are randomly colored. It is the shapes themselves that we are concerned about. The task for any model is to accurately detect all the triangles from the test set as anomalies.

We intend to use the artificial data as a benchmark for experimentation to establish the best model architecture and the best method for anomaly detection.

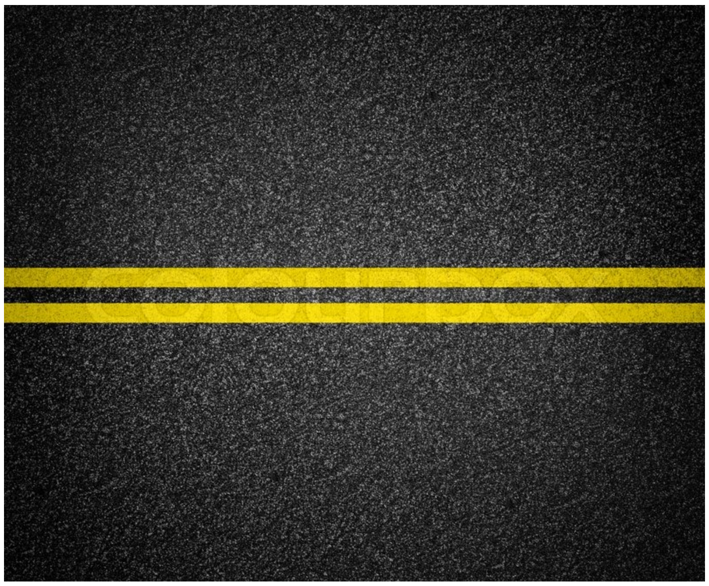


Figure 3–8 – Background for the artificial dataset.

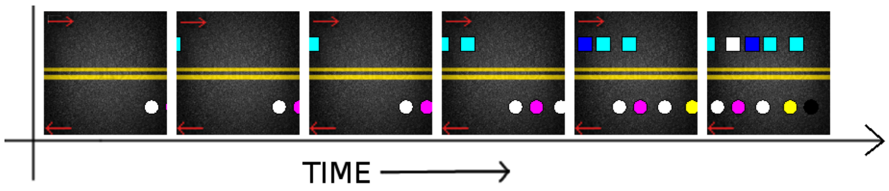


Figure 3–9 – Examples of training set images ordered in a timeline. The red arrows in the figures are representative of the direction in which the shapes move on the road.

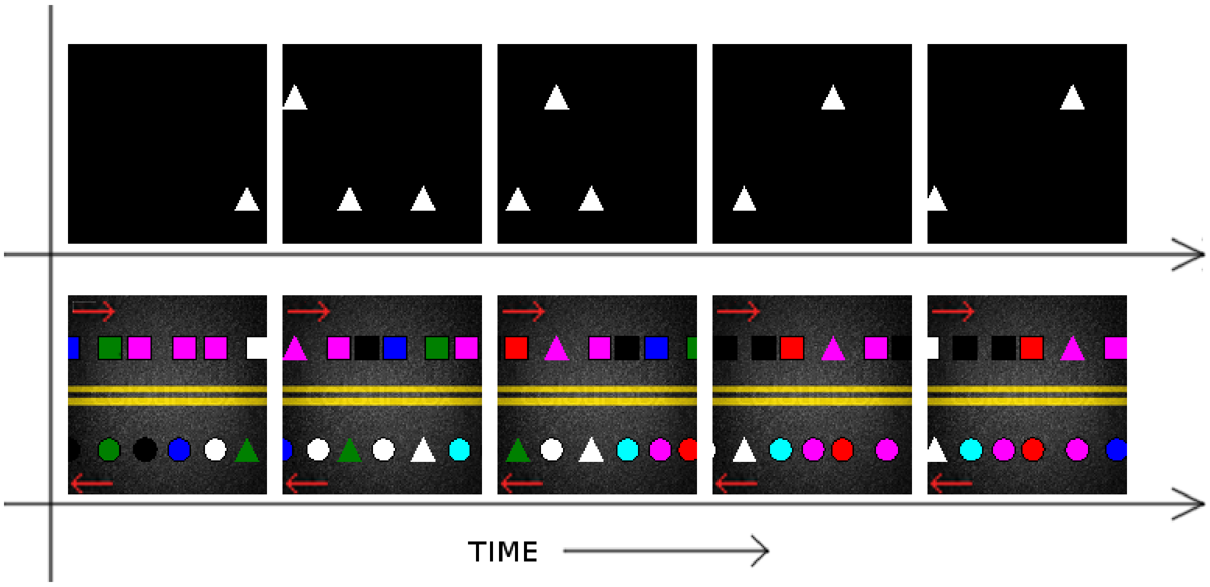


Figure 3–10 – Images from the test set of the artificial dataset. The red arrows in the figures are representative of the direction in which the shapes move on the road. Top row: Pixel level ground truth annotations for the corresponding test set images. Anomalous pixels are white in color. Bottom row: Examples of test set images ordered in a timeline.

3.3 Chapter summary

In this chapter, we discuss benchmark datasets available for anomaly detection. Following this, we establish that the benchmark datasets have a comparatively fewer number of data points to facilitate the use of deep learning for our problem. To overcome this constraint, we create an artificial dataset that is programmatically generated. This generates a big enough dataset to use deep learning for anomaly detection in videos. This chapter concludes with a study on how the dataset is created.

Chapter 4
The convolutional autoencoder

4.1 Convolutional neural network

To really understand the significance of the Convolutional Neural Network (CNN), we need to examine the history of neural networks and computer-vision.

4.1.1 A brief history of convolutional neural networks and computer-vision

It was in the year 1943, that McCulloch and Pitts [43] established that a mathematical approximation of neurons (perceptrons) can be combined to construct a Turing machine [27]. In 1958, Rosenblatt [54] showed that perceptrons could be trained to convergence if the data were represented effectively. This rise in interest in neural networks was subdued through a publication elucidating the limitations of neural networks by Minsky and Papert [47] in 1969. The research on neural networks lay dead for over a decade until Hinton *et al.* devised a learning algorithm for Boltzmann machines [3]. The widely used back-propagation algorithm was introduced shortly after by Rumelhart *et al.* [57] (1986). This wave of new research brought about the Neocognitron [22], a layered neural network which showed the capability of visual pattern recognition. Subsequently, in 1998, the first formally titled Convolutional Neural Network (CNN), trained using back-propagation was proven effective for document analysis by LeCun [36]. However, these networks were computationally intensive and required excessive computational resources to be trained. This aspect of the programs made them ultimately unusable for any practical applications during this time.

In parallel, computer-vision algorithms were becoming indispensable in a lot of practical applications. Hubel and Wiesel [28] showed the behaviour of receptive fields of single neurons in a cat's striate cortex in 1959. They demonstrated how a cat's brain reacted to differentials across its field of vision compared to static information. This fact was embraced by the field of computer-vision. It was shown that important features from a digital image could also be extracted by using pixel differential operations. This led to the introduction of the Sobel operator for edge detection in 1968 and the Canny algorithm for edge detection in 1986 [8]. Similarly, the Harris corner detector was introduced [24] in 1988, also using differential operations. Even though these algorithms performed very well in detecting edges and corners, they were still only base level features, and were not enough to extract a high-level understanding of the image. Local binary patterns [50] presented a notion of robust features extracted from an image to effectively distinguish textures. This was somewhat in parallel to the proposal of Histogram of Oriented Gradients (HOG) by McConnell in a patent application in 1986. The ideas popularized by Hubel and Weisel also inspired the Haar-like features [51] and the Viola-Jones algorithm for object detection [71]. The Viola-Jones algorithm was an immediate hit, and was widely applied in a lot of practical applications, as it was computationally perfunctory.

The key behind the HOG feature descriptor is that object shape and appearance could be summarized by evaluating the distribution of intensity gradients and their orientations. The image is divided into cells and sub-cells, and within each sub-cell, a histogram of pixel-intensity gradient orientations is assembled. The histograms from each sub-cell are concatenated to form the descriptor of that cell. These local histograms are subjected to contrast-normalization by scaling the histograms by the average local intensity in the image, providing better illumination invariance. Dalal and Triggs used this idea [15] to demonstrate good numbers in pedestrian detection from static images. These features, however, were all heavily subject to failure by a change in the scale of the images. Scale-Invariant Feature Transform (SIFT) was instituted by Lowe [39] to overcome these particular limitations. SIFT features are invariant to uniform scaling, orientation and illumination changes. SIFT proved to be robust in identifying objects even through clutter and partial occlusion. This inspired a more robust, speedier operator for image feature extraction called Speeded Up Robust Features (SURF) [6]. The SURF algorithm uses the idea of a pre-computed integral image to speed up detection and feature extraction. SURF features have been successfully employed in object recognition and localization, people and face detection, object tracking and image-matching.

As convolutional neural networks were hard to train, and computationally exhaustive, feature extractors like HOG, SIFT and SURF were used in high-level computer-vision tasks like object recognition, localization and image-matching. Typically, generalized robust features extracted from the images were used to train learning algorithms to perform the tasks. The Euclidean distance between features of a test-image and a feature database was minimized for image-matching, object recognition, and object localization.

There were some limitations within this method. Generalized features that seemed to perform well in some datasets were being used for feature extraction in all tasks. In convolutional neural networks, the feature selection is left to the model. This means that the back-propagation algorithm is used to train the CNN to pick the most useful features from the images so as to best achieve its training objective. The idea itself made a lot of sense, but CNNs had a large number of parameters to be tuned, and the lack of big datasets created problems with training bias and overfitting. Training of the convolutional neural networks were also plagued by technical difficulties owing to gradient saturation [23].

Any advance of CNN technology required a demonstration of its prowess and a solution for the gradient saturation problem [23]. AlexNet [34] satisfied both requirements by beating all other models in the ImageNet 2012 challenge. The ImageNet dataset was large enough to effectively train a CNN with a large number of parameters. Alex used rectified linear activation instead of sigmoid activation to solve the gradient saturation problem. Fast forward 6 years, CNNs are everywhere, from optical character recognition to self-driving cars and facial identification for securing personal devices.

4.2 Understanding a convolutional neural network

Like all deep neural networks, convolutional neural networks are hierarchical architectures. They are generally composed of convolutional layers, pooling layers and fully-connected layers. A typical convolutional network has alternating convolutional and pooling layers followed by fully connected layers. In the following sections, we will look at different aspects of the convolutional neural network as a step towards finally understanding the convolutional autoencoder.

4.2.1 The input

The input to a convolutional neural network is usually the normalized pixel intensities. For color images, traditionally, the input images will be 3-dimensional arrays of floating point values organized as : image height in pixels, image width in pixels, number of color channels. Color images are usually comprised of three channels whereas greyscale images consist of a single channel of pixel intensities.

4.2.2 The convolutional layers

For the scope of this work, we will examine only 2-dimensional convolutional layers. A 2-D convolutional layer (here-on referred to simply as a convolutional layer) has the following parameters : number of filters, kernel-size, strides, padding, and an activation.

1. Number of filters: The number of feature detectors in this layer. The input will be operated on (convolved) by these many feature detectors to generate feature maps.
2. Kernel-size: This specifies the height and width of the 2-D convolution window. In the literature, kernels are mainly 3x3 matrices.
3. Strides: Determines the strides of the convolution along the height and width. For example, a stride setting of (2,1) would imply that the convolution window moves 2 pixels along the height dimension after each row of convolution, and 1 pixel along the width after each convolution.
4. Padding: Whether or not to apply padding on the input to generate an output of the same height and width.
5. Activation : The activation function to be applied on each generated feature map to produce the filtered output, while also adding to the non-linearity of the model.

The role of each of the above can be illustrated with an example. Consider a convolutional layer with 32 filters, each of size 3x3, with strides (1,1) and padding with ReLU activation. Let the input to this layer be an RGB image of dimensions: (128,128,3). The convolutional layer operates as follows. The input image is convolved with each of the 32 kernels, generating 32 different 128x128 output images. The size of each output image is 128x128 because the strides were 1 pixel in each dimension, with padding. This results in the same spatial dimensions in the output as the input. Each of these output images are called *feature-maps.* Each pixel in the feature map is now filtered by an activation function. Since we use a rectified linear unit as activation in this illustration, values in the feature maps below zero are clipped to 0 while the positive values remain unchanged. Hence, the application of this convolutional layer on the input, results in a tensor of dimensions: 128x128x32. The output tensor is 32 activated feature-maps of size 128x128.

Some examples of convolutional kernels in a CNN and feature maps can be observed in Fig 4–1. Fig 4–1-1 shows 64 convolutional filters from the first convolutional layer of a VGGNet [64] trained on the Imagenet dataset. The filters are 3×3 in dimensions. Convolving an input image (Fig 4–1-2) with all the 64 filters result in 64 corresponding feature maps. The resultant feature maps are presented in Fig 4–1-3. We note that some filters in Fig 4–1-1 identically match the Haar wavelets discussed in Section 4.1.1.

4.2.3 The pooling layers

Just like the convolutional layers, only 2-dimensional pooling is studied in this thesis. Pooling layers are dimensionality reduction operators, acting as a filter to pass on fewer (more important or average) activations. It helps reduce the parameters of the network and hence enables a drop in overfitting. Some common pooling operators include spatial max-pooling and spatial average-pooling. The pooling layers have only one parameter, the size of the pooling operator. These are the factors by which to downscale (vertical, horizontal) the feature maps. For example (2, 2) will halve the input activations in both spatial dimensions, hence discarding 75% of all activations from the preceding layer.

1. <u>Spatial max pooling</u>: In each pooling step, the maximum activation from the pooling windows in each feature map is passed on, while the other values are dropped.

2. Spatial average pooling: In each pooling step, the average activation from the pooling windows in each feature map is passed on, while the other values are dropped.

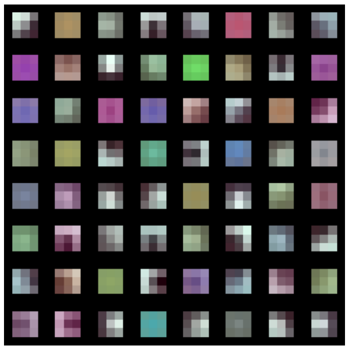
Pooling operations are illustrated in Fig 4–2.

4.2.4 The fully connected layers

After all convolutional layers, the final activations are concatenated and passed on to the fully connected layers. Fully connected layers are densely connected neural network layers, which act as a transformation step from the image-feature domain to the target domain. If the target is classification, the fully connected layers act as a feature transformation layer, converting the features selected from the images to features that help achieve classification. On the other hand, if the target is regression, the fully connected layers act as a feature transformation layer to convert the image-features into features that enable regression.

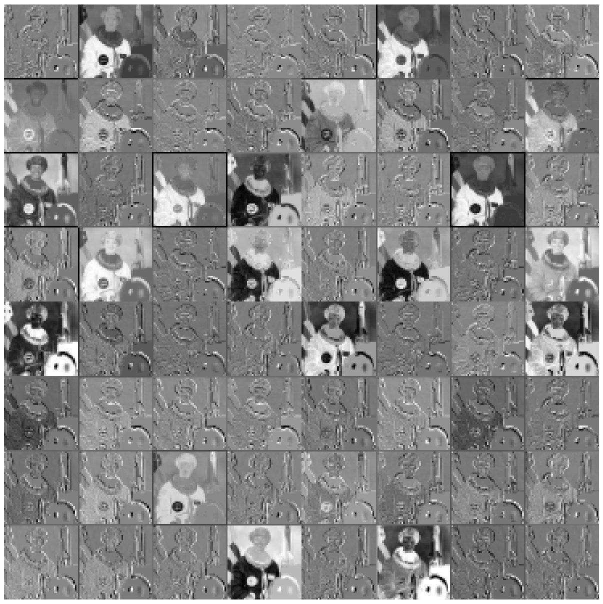
Consider a CNN for classification that takes an input image of size 64x64x3, and the final fully connected layer before the output layer is 1000 units. In this model, each image of 12288 pixels is converted to a vector of 1000 features for the supervised task. In other words, the CNN essentially acts as a feature descriptor that takes an image as input and encodes it into a vector of features.

A commonly used convolutional neural network architecture can be seen in Fig 4–3.



(1) Convolutional filters from the first convolutional layer of a trained VGGNet [64].

(2) The input image on which the convolutional filters are applied.



(3) Feature maps obtained from applying the convolutional filters on the input image. 64 feature maps for the corresponding 64 filters from (1).

Figure 4–1 – Convolutional filters of a trained VGGNet and feature maps from the first layer.

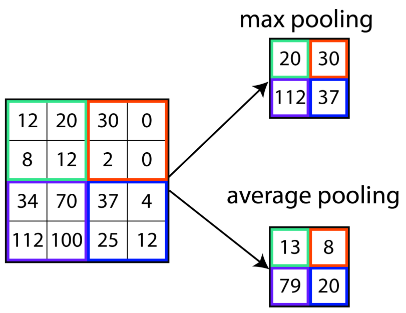


Figure 4–2 – An illustration of pooling operations.

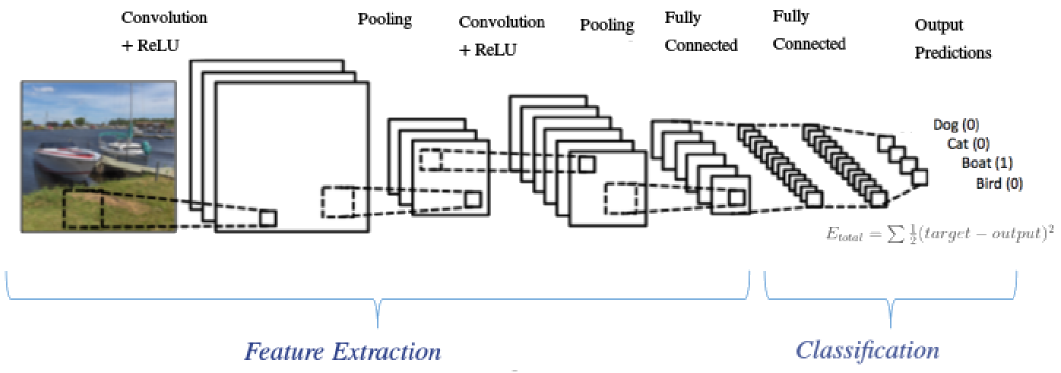


Figure 4–3 – An illustration of a convolutional neural network used for image classification. Image based on [1].

4.3 Convolutional autoencoder

Convolutional neural networks are ordinarily used in supervised tasks, modelling a function $f(\theta) : X \rightarrow Y$. On the other hand, convolutional autoencoders are used to model a function $f(\theta) : X \rightarrow X$,i.e to reconstruct the input using a parametric model. The autoencoder is ideally a data compression technique, where high-dimensional data are reduced to a lower dimensional feature space, while still retaining enough information to reconstruct the high dimensional input.

A convolutional autoencoder has two parts, an encoder and a decoder. The encoder has architecture similar to a normal convolutional neural network, mapping images to a low-dimensional feature space. The encoder is defined by $E(\gamma) : X \in \mathcal{R}^N \rightarrow Y \in \mathcal{R}^n \mid N > n$. A decoder works in reverse, using transposed convolution [81] and upsampling layers, and is defined by $D(\phi) : Y \in \mathcal{R}^n \rightarrow X \in \mathcal{R}^N \mid N > n$. A convolutional autoencoder can then be seen as an encoder, followed by a decoder, such that the encoded features characterize the input data in a lower dimensional feature space. Fig 4–4 depicts a typical convolutional autoencoder.

Building a convolutional autoencoder can be tricky, since the model might just copy the input to the output, hence delivering 100% performance. This can be constrained by making sure that the dimensionality of the encoder output is much smaller than the dimensionality of the input.

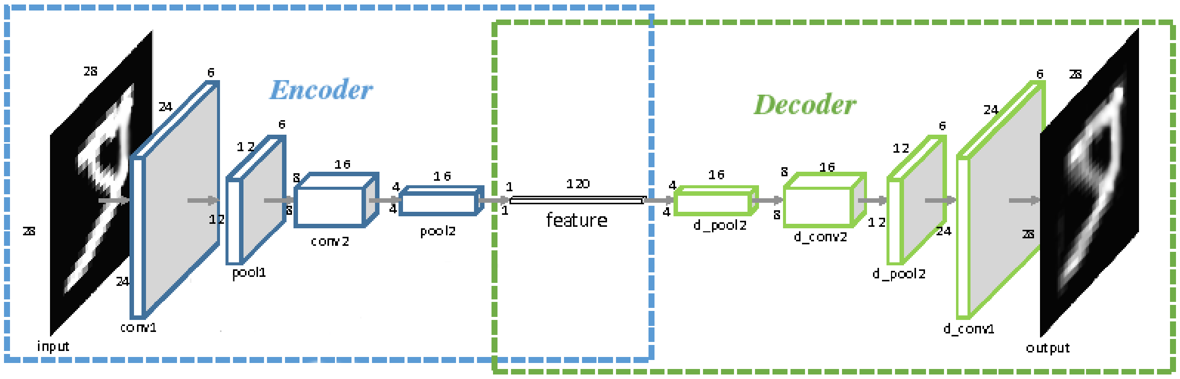


Figure 4–4 – An illustration of a convolutional autoencoder.

A model for generalized anomaly detection, ideally formulated to understand the nature of the scene being surveyed, should be able to summarize the scene in a *lower* dimensional feature space, which can be subjected to analysis, such that an anomalous event can be distinguished from the normal events in the scene. A convolutional autoencoder accurately fits the description of such a model. This makes it a top candidate for study in this work.

4.4 Chapter summary

This chapter is aimed at generating a good understanding of the convolutional autoencoder. Hence, the initial part of the chapter examines a brief history of the classical computer-vision techniques and convolutional neural networks, before exploring convolutional neural networks in detail. Finally, everything is brought together to understand the convolutional autoencoder as a candidate for performing anomaly detection in videos.

Chapter 5
Data preparation

5.1 Raw data

A static surveillance video camera generates and stores data in the following way. The scene is sampled with a set frequency known as a sampling rate. If the sampling rate is 30 frames per second, the camera samples the scene 30 times every second and sends the frames to a video encoder. These series of frames usually contain tremendous amounts of spatial and temporal redundancy. A video encoder attempts to compress the frames to reduce the redundancy and store the information in a more compact fashion, in the form of a video file. When the video is played back, a video decoder reads the encoded video file into a continuous stream of frames fit for human viewing.

Since our model is built on the foundation of feature extraction from images using convolutions, we cannot operate directly on the video file. Consequently, the first step in preparing the data is to use a decoder to extract sequential image frames from the video file.

5.2 Frames

Say the available surveillance footage starts at time $t = 0$ *seconds* and ends at time $t = N$ *seconds*. N is typically a considerable number, owing to the nature of scene surveillance. Scenes are usually subject to surveillance over long periods of time. When this footage is converted to a continuous stream of image frames for analysis, the data matrix containing all of these frames would be of dimensions: (image height, image width, N*sampling rate, channels). Consider an example of surveillance footage featuring a scene in greyscale, of resolution $256 * 256$, from a camera with sampling rate 30 frames per second, for a week. A week is comprised of 604800 seconds. Therefore by assignment, $N = 604800$. The video-cuboid data matrix of all the frames would be of dimensions : (256,256,604800*30,1) = (256,256,18144000,1).

This is a colossal amount of data to be modelled by an autoencoder directly and hence we need to split the data-matrix into observations of smaller size.

5.3 Video units

Splitting a video-stream into smaller parts can be a difficult task. The simplest choice is to divide the data into the existing component frames. We could train a convolutional autoencoder by treating each frame as a separate observation, but this would rid the model of any temporal information about the scene. It is reasonable to assume that it is necessary to investigate the dynamics in a scene in order to detect anomalies. Therefore any characterization of the scene should understand the temporal attributes of the scene as well. This would mean that every observation used to train the convolutional autoencoder must contain multiple frames such that the temporal disposition of the scene is captured.

As a solution, we may decide that we can divide the video stream into events. Unlike a movie, where each event can be thought of as a continuous shot from the same camera-angle, it is not straight-forward to split a surveillance stream into events. A surveillance stream is usually a continuous shot of the same scene without changing the camera focus. Motion can occur in any part of the frame and multiple events can take place in parallel. For example, a scene overlooking a road can have a car moving in one direction and a bicycle moving in another direction at the same time. Moreover, the time taken by the car to move across the field of vision is much lower than the time taken by the bicycle if they are both covering the same average distance.

In order to avoid presumptions in the process of scene modelling, we can forego the human understanding of an event and sub-sample the data-matrix into standard observations of uniform length. Most of these observations will not contain anything interesting. Hence the term event is a little confusing as we are merely describing a time interval in each observation. Thus, each of these observations are called *units*. The length of time in each unit should be long enough for the model to get a temporal sense of the scene from each observation, but short enough to facilitate the learning of the observation by an autoencoder.

In this thesis, this time is set to be 5 seconds (5×30 frames if the video is sampled at 30 fps) as it satisfies the above requirement for all the datasets we consider. Hence, the full video data-matrix is densely sub-sampled into units comprising of frames encapsulating 5 seconds of footage. Each unit is sampled with a stride of 1 second. For example, a video of 6 seconds will be sub-sampled into two units; the first unit spanning from $t = 1\ s$ to $t = 5\ s$, and the second unit spanning from $t = 2\ s$ to $t = 6\ s$.

Each input to the convolutional autoencoder will now be of dimensions: (image-height,image-width,150,number of channels).

5.4 Localization

Recalling the objectives of this work from Section 1.2.3, we wish to both detect anomalous events and localize their occurrence in the spatial extent of the frame. To facilitate this, the convolutional autoencoder can be trained on observations that are spatially sub-sampled from each unit. This means we can analyze the observations from different locations in the frame in every unit and classify each observation as an anomaly or not. The spatial sub-sampling window must be large enough to capture the spatial identity of each observation, but small enough to have a good resolution in locating the anomaly. The size of the window will hence be proportional to the frame size. This sub-sampling should be done using a stride value less than the size of the sub-sampling window itself. The stride is the number of pixels the sampling window is moved to extract the next observation (This stride is similar to the stride in convolution operation). Since each of the *units* have been spatially sub-sampled, each *sample* can now be referred to as a *spatio-temporal cuboid.*

The process of sub-sampling the data is illustrated in Fig 5–1.

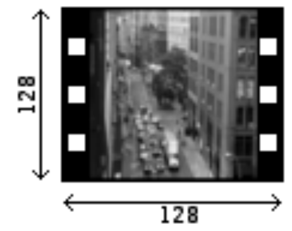
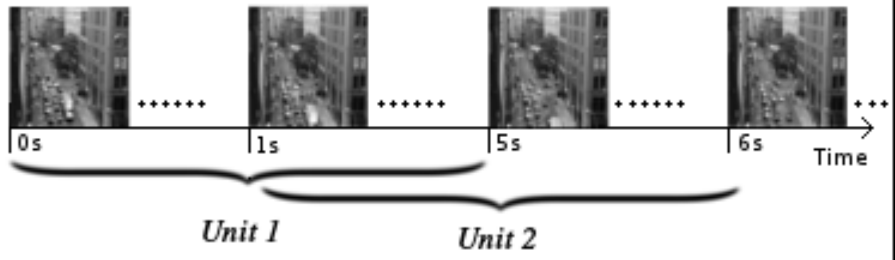
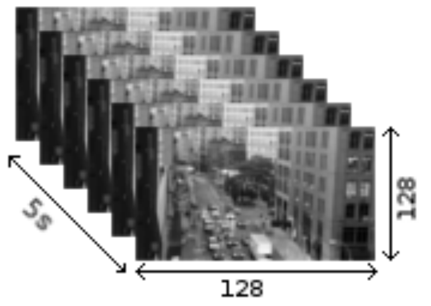
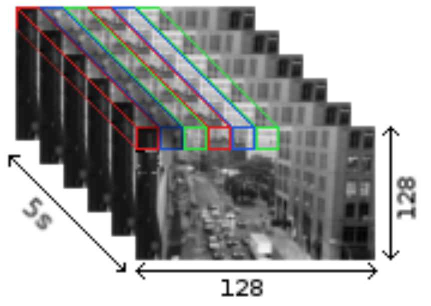
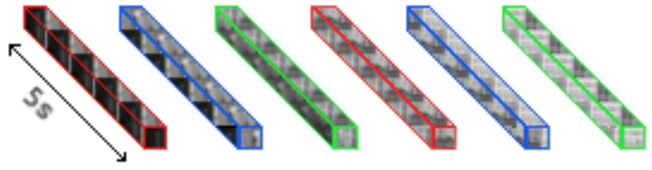
Input:	
Sequential Frames:	
Dense subsampling into units:	
Collect units into temporal cuboids: *(Only unit 1 illustrated)*	
Spatial subsampling of each unit:	
Spatio-temporal cuboids:	

Figure 5–1 – Illustration of the data-subsampling to create spatio-temporal cuboids.

5.5 Normalization

The anomalous events, if any, will be in the dynamics of scene, as explained above. Hence, it is advantageous to make our convolutional autoencoder focus only on the dynamic nature of the scene. Once the data are completely sub-sampled, the mean frame of each spatio-temporal cuboid evaluated and subtracted from each frame in the cuboid. This removes the static information from each spatio-temporal cuboid and it is left with the dynamic components only. Each of these mean-subtracted cuboids is now rescaled to lie in the range [0,1]. Normalization of the cuboid assists in the convergence of the model.

5.6 Summary

This chapter discusses the sampling of the video data into spatio-temporal cuboids. The video is first converted into sequential frames. The frames are densely sub-sampled into units spanning 5 seconds each with a stride of 1 second. Each unit is then spatially subsampled with a sampling window to obtain spatio-temporal cuboids. Each spatio-temporal cuboid is now normalized by subtracting the mean frame of the cuboid from each of the component frames. The normalized cuboids are then rescaled to lie in the range [0,1]. All the spatio-temporal cuboids are then compiled into a dataset to train the convolutional autoencoder.

Chapter 6
The deep convolutional autoencoder

As concluded earlier, the video surveillance data from a static camera is used to train a deep convolutional autoencoder. This chapter is devoted to understanding the convolutional autoencoder used for our solution. From section 4.3, it is evident that all convolutional autoencoders are composed of two blocks, an encoding block and the decoding block. We will discuss both of these parts sequentially, in view of generating a complete understanding of the model.

6.1 Encoder

The encoder is a function of the form $E(\gamma) : X \in \mathcal{R}^N \rightarrow Y \in \mathcal{R}^n \mid N > n$. It is essentially a dimensionality reduction transformation of the input. The encoder receives a spatio-temporal cuboid of dimensions : (cuboid-height, cuboid-width, frames, channels) as input and transforms it into a vector of n features (the encoding), such that this vector of n features can be used by the decoder to reconstruct the spatio-temporal cuboid.

6.1.1 Encoder architecture

The architecture of the encoder used in this work is illustrated in Fig 6–1-(1). It starts with an input layer and the output is a flattening layer, that aggregates the output into an n-dimensional vector. There are 4 convolutional layers arranged sequentially.

Each convolutional layer is preceded by a Gaussian noise layer to add noise to the input. The noise layer adds small perturbations to the input so that the convolutional autoencoder cannot simply memorize the best transformation of each input. The model is forced to learn the structural and characteristic features of the input to perform the transformation [70]. Due to the random noise added to each input, the model never sees the same exact input more than once, ensuring that the model learns the characteristics of the distribution of the inputs rather than each individual input cuboid.

The outputs of each convolutional layer (feature maps) are subjected to dropout [66], a technique proven to reduce over-fitting by eliminating co-dependence between feature maps. The idea of dropout is simple. When the model is being trained, a pre-set percentage of feature maps are randomly selected to be dropped out so that the model has to learn from a different set of feature maps each time. This constrains the model to learn features that are independently stronger.

The feature maps are activated by a leaky rectified linear unit layer [76] and subject to batch normalization [29] to keep the co-variance shift of the features in check. The normalized features are then filtered by average-pooling to reduce the dimensions of the feature-maps in half. The output of the last convolutional layer is activated, normalized and flattened, to return a single vector of n dimensions.

6.2 Decoder

The decoder is a function of the form $D(\phi) : Y \in \mathcal{R}^n \rightarrow X \in \mathcal{R}^N \mid N > n$. It essentially operates in reverse of the encoder. The decoder receives a vector of n features as input and it is transformed back into the spatio-temporal cuboid of dimensions: (cuboid-height, cuboid-width, frames, channels) as output.

6.2.1 Decoder architecture

The architecture of the decoder used in this work is illustrated in Fig 6–1-(2). It starts with an input layer and the output is a convolutional layer. The input is an n-dimensional vector and the output is the spatio-temporal cuboid.

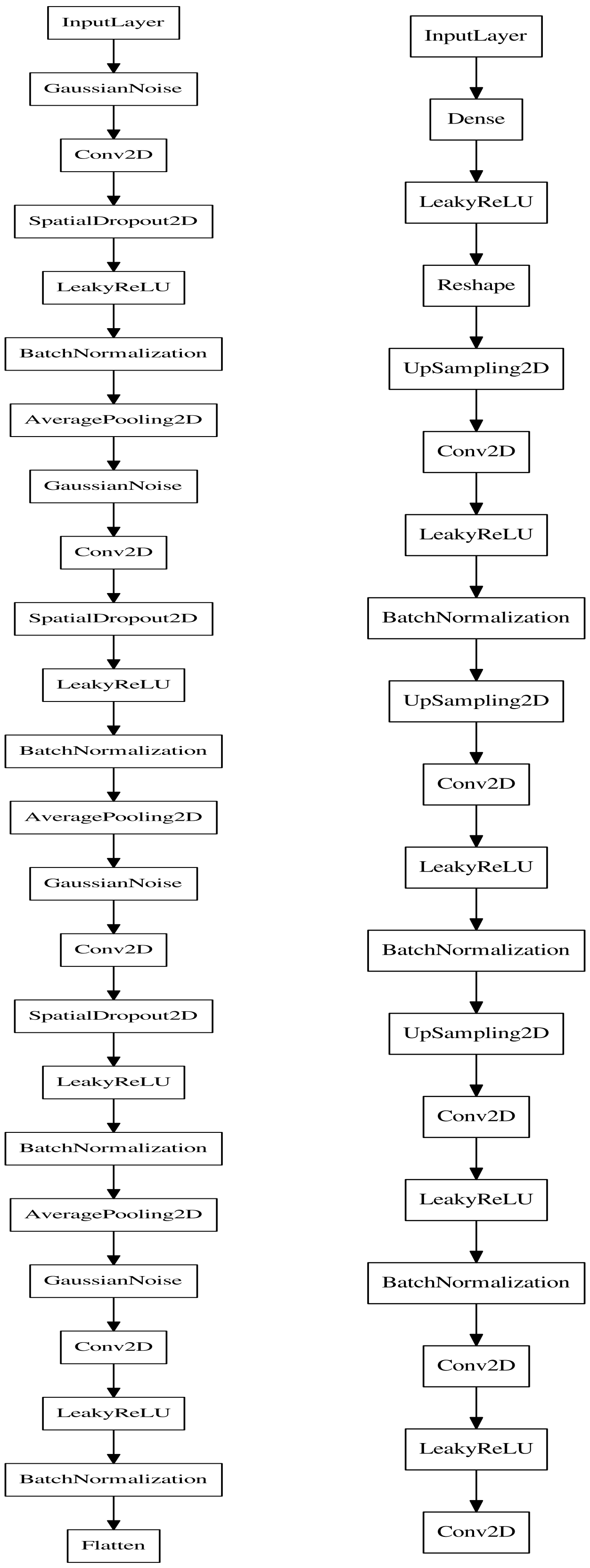
The first operation is a fully connected layer of neurons that transforms the n-dimensional vector into a higher-dimensional vector which can then be reshaped into the feature map domain. The output of the reshape layer will be of dimensions: (height, width, feature-maps). From this point, we can use up-sampling and convolutions to reach the target spatio-temporal cuboid.

There are 5 convolutional layers in the decoder architecture. Except for the final convolutional layer, all convolutional layers are preceded by an upsampling layer that uses bilinear interpolation to double the feature-map size in both spatial dimensions. The up-sampled feature-maps are then passed to the convolutional layers. The output of the convolutional layers, just like in the encoder, are activated by a leaky rectified linear unit. Similarly, the activated feature maps of the decoder are also batch-normalized.

The final convolutional layer is used only to transform the feature-maps to the frames. It converts the feature-maps of dimensions: (cuboid-height, cuboid-width, feature-maps) to the final spatio-temporal cuboid output of dimensions: (cuboid-height, cuboid-width, frames, channels).

6.3 The convolutional auto-encoder

The autoencoder is a function of the form $A(\gamma, \phi) : X \in \mathcal{R}^N \rightarrow X \in \mathcal{R}^N$. The convolutional auto-encoder is the encoder and decoder in series. The input and output of the auto-encoder are the spatio-temporal cuboids. Since we are not training a generative model to generate new samples from the distribution, we only need the encodings of each spatio-temopral cuboid in the low-dimensional feature space to evaluate its anomaly score. The decoder is a necessary evil so that the encodings are representative of the input cuboid. The parameters of the auto-encoder (γ and ϕ) are trained to reconstruct the input at the output using gradient descent and back-propagation. The specific details for the training process will be discussed in the next section. It should be noted that noise layers and dropout layers are present only in the encoder. This is to make sure that the decoder can always reliably reconstruct the encoding while training.



(1) Encoder (2) Decoder

Figure 6–1 – Encoder and decoder sections of the convolutional autoencoder.

6.4 Chapter summary

This chapter is a discussion about the detailed convolutional autoencoder architecture used in this work. The encoder and decoder architectures are studied in detail to elucidate the purpose of all the layers.

Chapter 7
Training

7.1 Overview of the objectives for this thesis

We will now recall the objectives of this thesis before the discussion of the training method. This will help in connecting the training steps to the objectives we need to accomplish. As described earlier, we are trying to train a convolutional autoencoder on spatio-temporal cuboids extracted from a surveillance video. We will then use the trained model to detect anomalous events in the surveillance footage captured in the future.

The convolutional autoencoder encodes the high-dimensional spatio-temporal cuboids into lower-dimensional vectors. The vectors are analyzed to evaluate whether the cuboid belongs to an anomalous event. It is also imperative to note that the spatio-temporal cuboids used to train the convolutional autoencoder are all normal cuboids with no anomalies.

7.1.1 What is an anomaly?

Probably, the most difficult aspect of this research was how to define the meaning of "anomaly" from the point of view of mathematics. The decision to use a convolutional-autoencoder was straightforward. The convolutional autoencoder is only used to reduce the dimensionality of each cuboid so that we could reliably analyze each low-dimensional encoding instead. However, what analysis needs to be performed on the encodings to classify the corresponding cuboids as anomalous or normal? What needs to be done in the training process to enable the analysis?

As concluded in the initial sections, anomalous events are events with a low probability of occurrence. In terms of a probability density function, these will be the samples where the probability density is extremely low. An illustration to this effect can be observed in Fig 7–1. It is to be noted that the probability density at a point is directly related to how often the sample might be observed if an infinite number of random samples were drawn from the distribution. We would like to translate this idea into a method that can be employed to classify the encoding from each spatio-temporal cuboid. In simple terms, we wish to extract from each low-dimensional encoding, a measure of the probability of its occurrence in the distribution. The farther it is from the densest part of the distribution (the tree), the more probable it is an anomaly, than not.

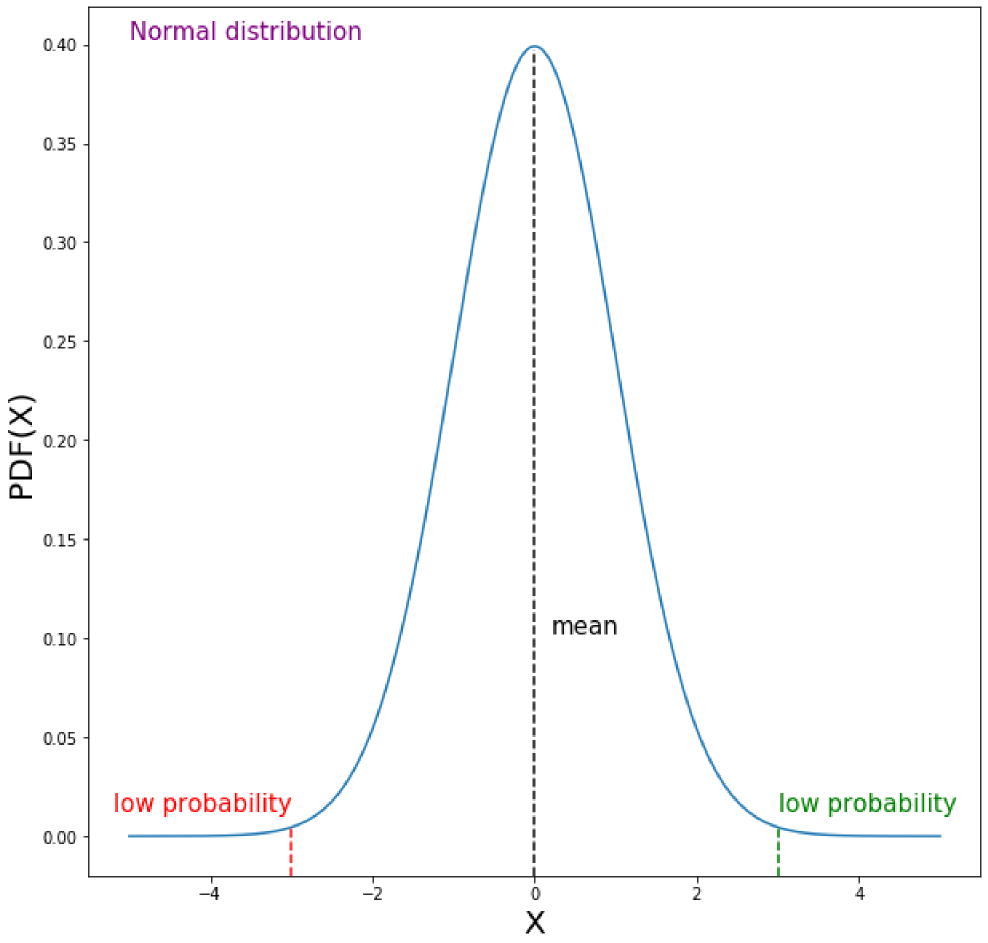


Figure 7–1 – Illustration of the anomalies in terms of probability density function from a normal distribution.

7.2 The formulation of training

Armed with the foresight of what needs to be done with the encodings to perform the classification, we now need to train the convolutional autoencoder to enable this. In this section, we will examine the formulation of the training objective which will help us achieve our objective of anomaly detection.

To see how far an encoding lies from the mean of a fitted distribution (equivalent to a cluster centroid), the encodings must lie in a space that is favourable for clustering and for fitting multivariate distributions to it.

As far as clustering is concerned, K-Means [38] is the most popular algorithm. Given a dataset $\{x_i\}_{i=1,....N}, x_i \in \mathcal{R}^M$, clustering aims to separate N data samples into K categories using the following equation,

$$\min_{\boldsymbol{M}\in\mathbb{R}^{M\times K},\{\boldsymbol{s}_i\in\mathbb{R}^K\}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{M}\boldsymbol{s}_i\|_2^2 \tag{7.1}$$
$$\text{s.t.} \quad s_{j,i} \in \{0,1\},\ \mathbf{1}^T\boldsymbol{s}_i = 1 \ \ \forall i,j,$$

where $\boldsymbol{s}_i$ is the assignment vector of data point i, which has only one non-zero element. We let $s_{j,i}$ denote the jth element of $\boldsymbol{s}_i$ and the kth column of $\boldsymbol{M}$, i.e., $\boldsymbol{m}_k$, denotes the centroid of the kth cluster.

In general, the K-Means algorithm works very well when the data-samples are spherically scattered (same variance in all dimensions) around the centroids in the multivariate feature space. These types of datasets are called K-Means friendly datasets. This situation can be demonstrated in a 2-dimensional space very easily. In Fig 7–2, the top row illustrates a K-Means friendly dataset that has been correctly clustered by the K-Means algorithm. The left column of the figure represents the original memberships of the distribution (by color) and the right column of the figure represents the memberships obtained by K-Means algorithm. It can be observed that the anisotropic dataset in the bottom row is unsuitable for K-Means owing to the incorrect class-memberships obtained when employing K-Means.

High-dimensional multivariate data are generally not K-Means friendly. Since we are using a convolutional-autoencoder as a nonlinear transformation to obtain encodings of spatio-temporal cuboids in a multivariate feature space, we need to make sure that the transformation produces an approximately K-Means friendly space. To ensure this, we perform the clustering in parallel to the training of the convolutional autoencoder, using the method suggested in [78]. During training, a K-Means algorithm is performed to initialize a set of

K centroids $= \boldsymbol{M}$ on the encodings $\boldsymbol{E}(\gamma, X)$. These $\boldsymbol{M}$ centroids are then updated and used in the training objective function to guarantee that the multivariate feature space will favour clustering. To guarantee this, the convolutional autoencoder is to be trained with the following optimization objective in Eq:7.2 [1] .

$$
\begin{aligned}
&\min_{\gamma,\phi,M} \sum_{i=1}^{N} \left(\ell\left(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2} \left\| \boldsymbol{E}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i \right\|_2^2 \right) && (7.2)\\
&\text{s.t.}\;\; s_{j,i} \in \{0,1\},\; \mathbf{1}^T s_i = 1 \;\; \forall i, j, \\
&\quad \ell(\cdot) : \mathbb{R}^M \to \mathbb{R} = \text{reconstruction-loss} && (7.3)\\
&\quad \boldsymbol{D}(\phi) = \text{Decoder} && (7.4)\\
&\quad \boldsymbol{E}(\gamma) = \text{Encoder} && (7.5)\\
&\quad \boldsymbol{M} = \text{Means} && (7.6)\\
&\quad \boldsymbol{s} = \text{mean membership} && (7.7)\\
& && (7.8)
\end{aligned}
$$

The reconstruction loss $\ell(\cdot) : \mathbb{R}^M \to \mathbb{R}$ used in this work is the structural dissimilarity loss as described in [72] and [73]. The structural dissimilarity is a much smoother metric to evaluate the similarity between images when compared to the least-squares loss or cross-entropy loss and in general, produces much better reconstructions of spatio-temporal cuboids. This fact was verified in one of the many experiments done in this research.

[1] The training algorithm discussed in this section is inspired by [78].

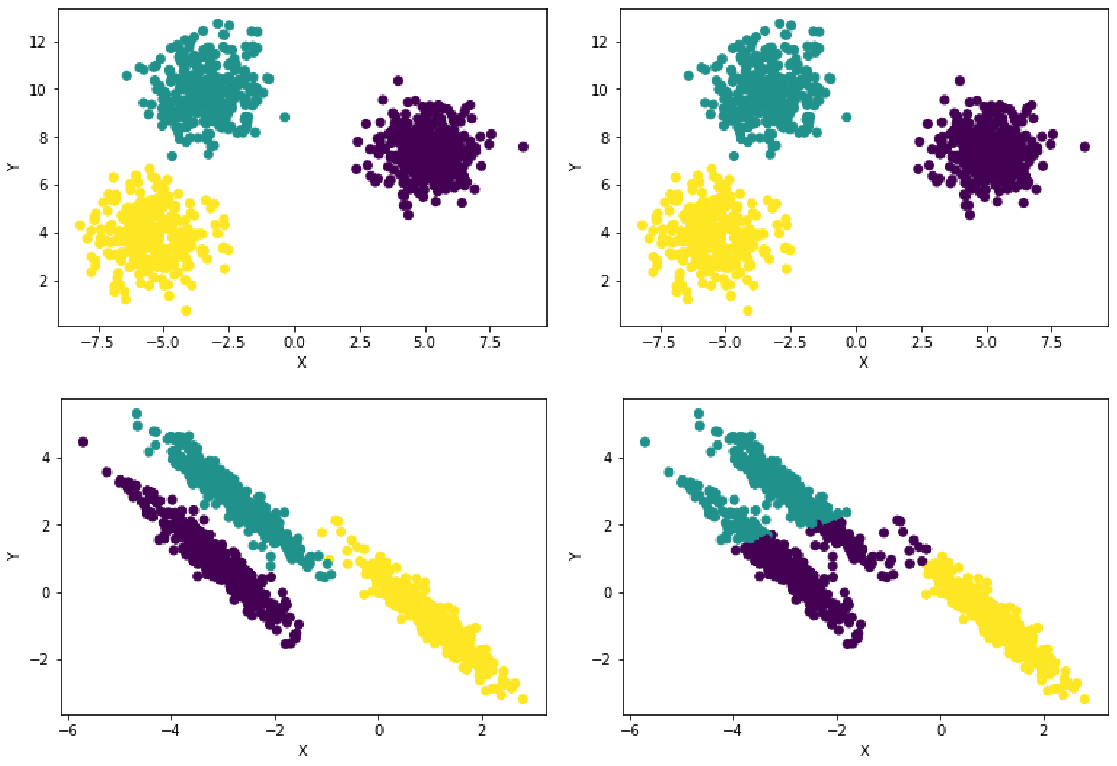


Figure 7–2 – Illustration of K-Means friendly datasets. Top row: K-Means friendly dataset. Bottom row: Anisotropic data unsuitable for K-Means. Left column: Dataset with clusters colored by original membership, Right column: Dataset with clusters colored by membership obtained from K-Means algorithm.

7.3 Training algorithm

Optimizing Eq. 7.2 is not a simple task [2] . The constraints and the cost function are both non-convex. There are multiple scalability issues that need to be considered. In this section, we describe a practical optimization procedure that includes an empirically effective initialization method, coupled with an alternating optimization algorithm.

7.3.1 Initialization

The training process starts by initializing the parameters γ and ϕ of the encoder and the decoder, respectively. For this initial pre-training stage, λ in equation 7.2 is set to zero. The resulting formulation to be optimized becomes:

$$\min_{\gamma,\phi,M} \sum_{i=1}^{N} \ell(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}_i)), \boldsymbol{x}_i) \tag{7.9}$$

$$\ell(\cdot) : \mathbb{R}^M \to \mathbb{R} = \text{reconstruction-loss} \tag{7.10}$$

$$\boldsymbol{D}(\phi) = \text{Decoder} \tag{7.11}$$

$$\boldsymbol{E}(\gamma) = \text{Encoder} \tag{7.12}$$

The gradient for minimizing this equation for one sample x_i is given by: $\nabla_\theta L^i = \frac{\partial \ell(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}_i)),\boldsymbol{x}_i)}{\partial \theta}$, where $\theta = (\gamma, \phi)$ is a collection of the network parameters. The network parameters can be updated by :

$$\theta \leftarrow \theta - \alpha \nabla_\theta L^i, \tag{7.13}$$

where α is the learning rate.

The parameters γ and ϕ are optimized using gradient descent for $\boldsymbol{T}$ epochs. This initialization makes sure that the encodings are in a space which enables accurate reconstruction so that the convolutional autoencoder does not learn trivial encodings.

[2] The training algorithm discussed in this section is inspired by [78].

7.3.2 Initialize means and memberships

After the first $\boldsymbol{T}$ epochs of training, a set of means $\boldsymbol{M}$ and mean memberships $\{\boldsymbol{s}\}$ are initialized. A set of K means $\boldsymbol{M}$ can be initialized by running the K-Means algorithm on the encodings $\boldsymbol{E}(X)$ until convergence. The mean memberships $\{\boldsymbol{s}\}$ are generated by setting

$$s_{j,i} \leftarrow \begin{cases} 1, & \text{if } j = \underset{k=\{1,\dots,K\}}{\arg\min} \left\| \boldsymbol{E}(\boldsymbol{x}_i) - \boldsymbol{m}_k \right\|_2, \\ 0, & \text{otherwise.} \end{cases} \tag{7.14}$$

7.3.3 Stochastic optimization with means update

A stochastic gradient descent algorithm using backpropagation cannot be used to jointly optimize $\gamma, \phi, \boldsymbol{M}$ and $\{\boldsymbol{s}\}$ since the mean memberships variable $\{\boldsymbol{s}\}$ is constrained on a discrete set, hence making the joint-objective non-differentiable. Hence, we need to alternate the optimization of $\boldsymbol{M}$, $\{\boldsymbol{s}\}$ and γ, ϕ.

This is done by optimizing 7.2 (w.r.t.) one of $\boldsymbol{M}$, $\{\boldsymbol{s}\}$ and γ, ϕ. while keeping the other two sets of variables fixed.

Updating the convolutional autoencoder

The parameters of the convolutional autoencoder $A(\gamma, \phi)$=$\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))$, γ and ϕ, is optimized while keeping $\boldsymbol{M}$, $\{\boldsymbol{s}\}$ fixed. This problem is similar to training a convolutional autoencoder using a regularization term. This regularization term penalizes the network more if it encodes a sample further away from its respective mean. This regularization ensures that the encodings lie in a clustering friendly space. To implement the gradient descent for updating the network parameters, we look at the problem formulation for this step in the alternating optimization procedure. For one data-point $\boldsymbol{x}_i$, the formulation is as follows:

$$\min_{\gamma,\phi} L^i = \ell\left(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2} \left\| \boldsymbol{E}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i \right\|_2^2. \tag{7.15}$$

The gradient of the above function over the network parameters is easily computable, i.e., $\nabla_\theta L^i = \frac{\partial \ell(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}_i)),\boldsymbol{x}_i)}{\partial \theta} + \lambda \frac{\partial \boldsymbol{E}(\boldsymbol{x}_i)}{\partial \theta}(\boldsymbol{E}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i)$, where $\theta = (\gamma, \phi)$ is a collection of the network parameters and the gradients $\frac{\partial \ell}{\partial \theta}$ and $\frac{\partial \boldsymbol{E}(\boldsymbol{x}_i)}{\partial \theta}$ can be calculated by back-propagation [57]. Then, the network parameters are updated by :

$$\theta \leftarrow \theta - \alpha \nabla_\theta L^i, \tag{7.16}$$

where $\alpha > 0$ is the learning rate for the stochastic gradient descent algorithm.

Updating the means

The matrix of cluster memberships $\{\boldsymbol{s}\}$ can be easily updated once M and the encoder parameters γ are fixed. The assignment of one sample $\{\boldsymbol{s}_i\}$ can be easily updated by :

$$s_{j,i} \leftarrow \begin{cases} 1, & \text{if } j = \underset{k=\{1,\ldots,K\}}{\arg\min} \left\| \boldsymbol{E}(\boldsymbol{x}_i) - \boldsymbol{m}_k \right\|_2, \\ 0, & \text{otherwise.} \end{cases} \tag{7.17}$$

After $\{\boldsymbol{s}_i\}$ and (ϕ, γ) are fixed, $\boldsymbol{M}$ can be updated in multiple ways. For instance, we may use $\boldsymbol{m}_k = (1/|\mathcal{P}_k^i|) \sum_{i \in \mathcal{P}_k^i} \boldsymbol{E}(\boldsymbol{x}_i)$, where $\mathcal{P}_k^i$ is the number of samples assigned to cluster k from the first sample to the current sample i. This method is quite intuitive, but it can cause a lot of issues for online updates, since the historical data (i.e., $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i$) can be far away from the actual cluster structure of the encodings. To overcome these issues, we can utilize the idea in [63].

We adaptively change the learning rate of updating, $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K$. Let us assume that the clusters are approximately equal in the number of data memberships. The foundation of the method is straightforward and intuitive. After updating $\boldsymbol{M}$ for a certain number of samples, the centroids of clusters with more members should be updated more smoothly while the other centroids are updated more aggressively.

To implement this, let p_k^i be the membership count of cluster k before handling the incoming sample $\boldsymbol{x}_i$. We now update $\boldsymbol{m}_k$ by the following gradient step:

$$\boldsymbol{m}_k \leftarrow \boldsymbol{m}_k - ({}^1\!/_{p_k^i}) \left(\boldsymbol{m}_k - \boldsymbol{E}(\boldsymbol{x}_i) \right) s_{k,i}, \tag{7.18}$$

where the gradient step size $1/p_k^i$ controls the learning rate.

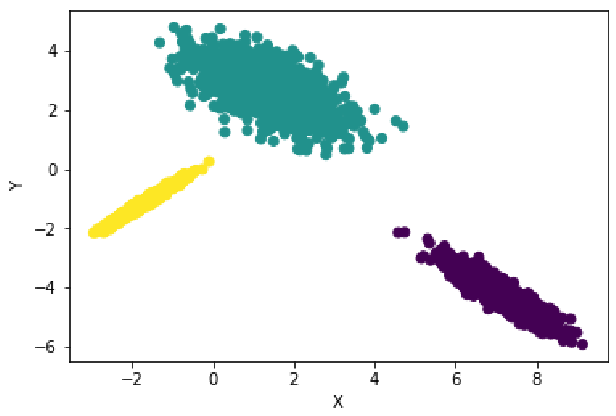
7.4 Feature space analysis

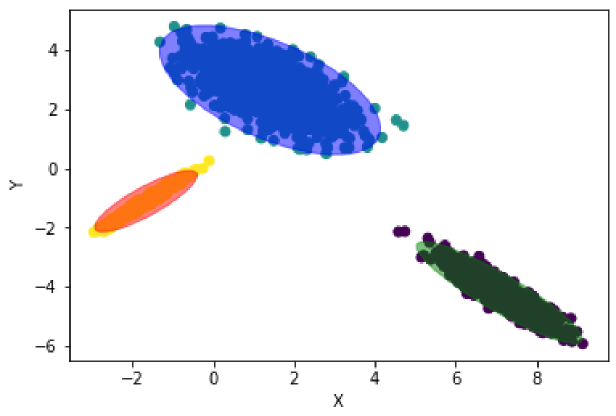
Once the training procedure is finished, we create a set of all the encodings of the training samples as given by the encoder of the convolutional autoencoder. The feature space is $F = E(X)$, where $E(\cdot)$ is the encoder and X is the set of spatio-temporal cuboids extracted from the video surveillance footage.

F is the feature space of the encodings from the training set. This is the distribution of normal features extracted from the surveillance videos. This feature space is now modelled as a multivariate distribution (Q) so that we can obtain a probability of the encodings of the spatio-temporal cuboids from the test footage. Each spatio-temporal cuboid from the test footage is encoded into the feature space and tested against the distribution Q. *The cuboid is classified as an anomaly if it has a very low probability of coming from Q.*

Since we initialized K means for clustering the encodings in the feature space, we fit a multivariate Gaussian mixture distribution model G [53] on F, with K Gaussians using expectation maximization [48]. An illustration of a Gaussian mixture model on a 2-D space can be seen in Fig 7–3. After the Gaussian mixture model is fitted to the data, we can evaluate the negative weighted log-probabilities for any point in the feature space from the Gaussian mixture model. As observed from Fig 7–3-(3), it can be observed that points far away from the mean (centroid) have very high negative log probability density.

The negative log probability density on a normal distribution is illustrated in Fig 7–4. As seen, the negative log probability is higher for low probability regions in the range. The negative log probability value is also exponentially related to the distance of the sample from the mean. Hence, after fitting a mixture of Gaussians on the feature space, we can use the negative log probability value as a metric that informs us how far the sample is from the mean.


(1) 2-D data.


(2) Gaussian mixture model with 3 Gaussians fitted on the data.

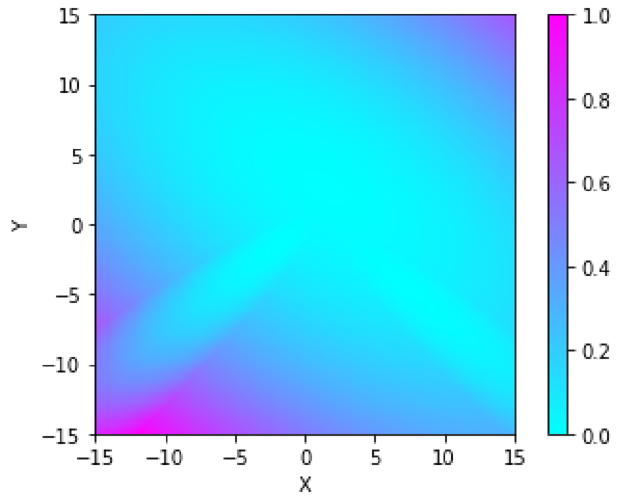
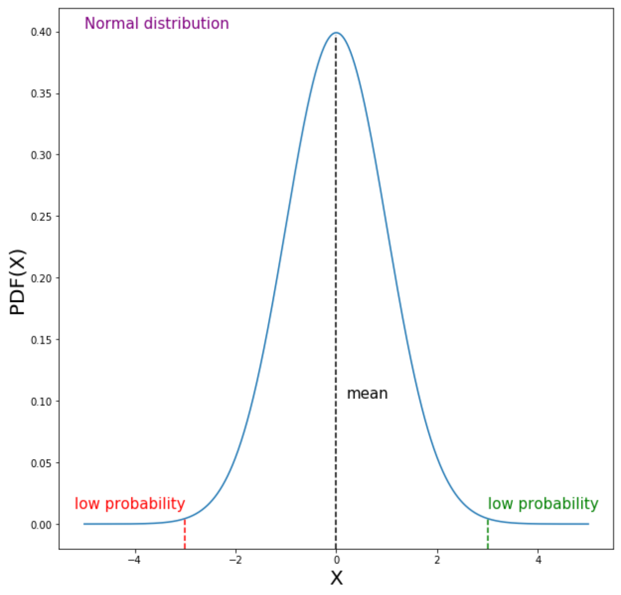
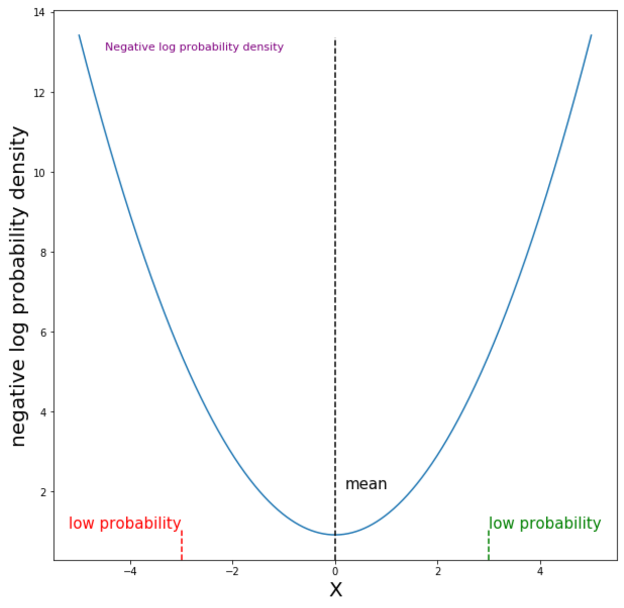

(3) Scaled negative log probability scores of the 2-d space on the fitted Gaussian mixture model.

Figure 7–3 – Illustration of Gaussian Mixture model on a 2-D feature space.



(1) Normal distribution.



(2) Negative log probability density function on the distribution.

Figure 7–4 – Illustration of negative log probability density on the normal distribution.

7.5 Summary of the training algorithm

The entire training algorithm can be summarized in Algorithm 1. One epoch is a network pass over all the training data samples.

Algorithm 1: Alternating gradient descent optimization.

```
1:  Initialization.
2:  Fix $\lambda = 0$.
3:  for $t = 1 : T$ epochs do
4:      Update network parameters by optimizing (7.13).
5:  end for
6:  Initialize $\boldsymbol{M}$ and $\boldsymbol{s}$ to start.
7:  Fix $\lambda = 1$.
8:  for $t = 1 : R$ epochs do
9:      Update network parameters by (7.16).
10:     Update assignment by (7.17).
11:     Update centroids by (7.18).
12: end for
13: Fit a Gaussian mixture model $\boldsymbol{G}$, with $\boldsymbol{K}$ Gaussians on $F = E(X)$.
```

Chapter 8
Results and analysis

In this section, we look at the proposed solution with a critical eye. The results from the proposed algorithm on all the datasets are presented and subjected to interpretive analysis.

8.1 Evaluation process

Prior to looking at the actual results, we review the process of evaluating the model in detail. This will provide a background for portraying the results.

Once the spatio-temporal cuboids extracted from the training videos are used to train the convolutional autoencoder using Algorithm 1, the evaluation process is initialized. An evaluation set is first assembled by extracting spatio-temporal cuboids from the test videos that contain the labelled anomalies. This set of spatio-temporal cuboids is called V. All the datasets considered to evaluate this work are ground-truth annotated. Hence, we can assemble a set of labels, which informs us of the anomalous nature of all members in V. This set of labels is called Y.

The high dimensional spatio-temporal cuboids $V \in \mathbb{R}^{\boldsymbol{N}}$, are converted to the lower dimensional encodings, $F_v \in \mathbb{R}^{\boldsymbol{n}}$, using the encoder $E(\gamma)$ from the trained convolutional autoencoder, such that :

$$F_v = E(V, \gamma) \tag{8.1}$$

The multivariate Gaussian mixture model G is now queried for each encoding f_v in the set F_v to determine which of the K Gaussians' f_v is a member of. Let g be the Gaussian to which f_v belongs to. The mean and covariance of the distribution g , namely, μ_g and Σ_g are used to evaluate the negative log probability density l for f_v in g, by employing the equation:

$$l = -\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma_g)}}\exp^{-\frac{1}{2}(f_v-\mu_g)^T\Sigma_g^{-1}(f_v-\mu_g)}\right) \tag{8.2}$$

$$= -\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma_g)}}\right) - log\left(\exp^{-\frac{1}{2}(f_v-\mu_g)^T\Sigma_g^{-1}(f_v-\mu_g)}\right) \tag{8.3}$$

$$= -\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma_g)}}\right) + \frac{1}{2}(f_v-\mu_g)^T\Sigma_g^{-1}(f_v-\mu_g) \tag{8.4}$$

Since the first term in equation 8.4 is a constant for g, we can conlcude that:

$$l \propto (f_v-\mu_g)^T\Sigma_g^{-1}(f_v-\mu_g) \tag{8.5}$$

Now, for comparing the negative log probability density of two samples, we can use the positive square root of the negative log probability density as well (monotonicity).

$$\sqrt{l} \propto \sqrt{(f_v-\mu_g)^T\Sigma_g^{-1}(f_v-\mu_g)} \tag{8.6}$$

Incidentally, the RHS of equation 8.6 is equal to the Mahalanobis distance [42], a measure that "intuitively" evaluates the number of standard deviations from f_v to μ_g. Hence, we can use the Mahalanobis distance as a measure of the anomaly score *for each spatio-temporal cuboid*. In other words, the greater the Mahalanobis distance, the higher the probability that the spatio-temporal cuboid is an anomaly.

An illustration of Mahalanobis distances evaluated on a 1-D zero mean normal distribution can be observed in Fig 8–1. The Mahalanobis distances from all the encodings in F_v are evaluated and collected in a set M. M is then subjected to thresholding. We can use the set of labels Y and the thresholded labels Y_{th} to evaluate the ability of the model to classify the spatio-temporal cuboids V as anomalies or not, based on M.

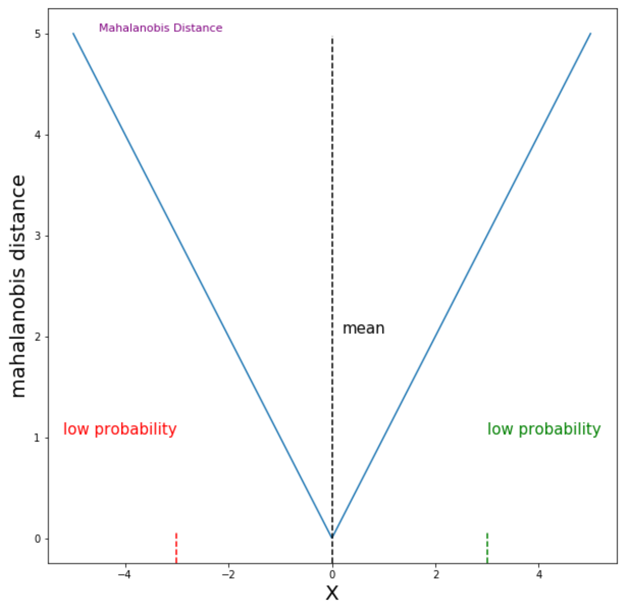


Figure 8–1 – Illustration of the Mahalanobis distance evaluated on a normal distribution.

8.1.1 Training setup

The training and validation for all the datasets were done using an 8-core Intel Xeon CPU E5-1620 v3 @ 3.50GHz and a 12GB GeForce GTX Titan GPU [1] . The code is written in Python [56] and the training is implemented using Keras [11] and Tensorflow [2]

[1] The python code can be found here: https://github.com/BitFloyd/Anomaly-Detection-DenseCuboid

8.2 Artificial dataset

The artificial dataset (described in Section 3.2) was developed in an effort to make high-volume ground-truth annotated data available for anomaly detection in videos by using deep learning. The results presented here, have been obtained from training the model using 60,000 frames and evaluating it on 6,500 frames. There are 100 anomalous events recorded in all of the test frames. Let us recollect that the anomalous events we are seeking are triangles moving across the scene (The training set consists only of squares and circles).

The distribution of the Mahalanobis distance metrics (M, as in Section 8.1) is captured in Fig 8–2. The Mahalanobis distance scores of the anomalous spatio-temporal cuboids are plotted in red, while the scores of normal cuboids are plotted in green. This color distinction is based on the true labels Y and not the thresholded labels Y_{th}. The threshold line drawn in blue is the threshold which gives us the maximum F1 score based on evaluating Y_{th} against Y.

Evidently, and as expected, the anomalous cuboids have much higher Mahalanobis distance scores than the normal cuboids. To evaluate the efficacy of this score as a classification metric, we can vary the threshold across the range of the score distribution, and calculate classification scores such as accuracy, precision, recall and F1-score at each point. The ROC curve obtained from this exercise is illustrated in Fig 8–3. Using the Mahalanobis distance as the anomaly score for spatio-temporal cuboids , we can obtain a perfect classifier for frame level anomaly detection (ROC curve for frame level anomaly detection is depicted in Fig 8–4).

It should be noted that these results are an evaluation of *how good the model can be* as a classifier for each of the datasets. In other words, we are evaluating the performance of M (extracted from the low-dimensional encodings of the spatio-temporal cuboids) as a metric to classify the anomalous cuboids from the normal cuboids.

8.2.1 Analysis

Using our solution on the artificial video dataset gives us a near perfect classifier for detecting and localizing the anomalies. The classification metrics of the model for the artificial video dataset is given in Table 8–1.

Some frames from the evaluation set with the spatio-temporal cuboids classified as anomalies are represented in Figure 8–5. The spatio-temporal cuboids are classified at the threshold where the evaluation gives us the maximum F1-score. Cuboids with a score lower than the threshold are classified as normal cuboids while a higher score is deemed to be anomalous.

Let us now take a moment to understand the classified spatio-temporal cuboids represented in Figure 8–5. Recall that the existence of triangles are anomalies in the artificial dataset. The figure depicts three different frames taken from the test set. The red hue is for correctly classified spatio-temporal cuboids. The green hue is for the misclassified spatio-temporal cuboids. If two cuboids are spatially overlapping in a frame, they are merged to form a bigger rectangle. Hence the difference in size for the green and red rectangles.

As seen, the misclassified cuboids are only at the edges of turquoise circles. This is because the pixelated corner of the circle resembles part of an edge of a triangle. This could have been avoided if we had used higher resolution images to train the model. However, higher resolution images in training would multiply the computational complexity of the algorithm, resulting in a much higher training and evaluation time.

The hyperparameters for training the model on the artificial dataset is given in Table 8–2.

Maximum Accuracy	**Maximum F1-Score**
98.4	83.21

Table 8–1 – Accuracy and F1-score metric for artificial dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	32
K (Number of Gaussians, Number of cluster centroids)	20
R (Number of epochs)	25
Threshold of Mahalanobis distance score for maximum F1 score	89.06
λ	0.001

Table 8–2 – Hyperparameters for evaluating the model on the artificial dataset.

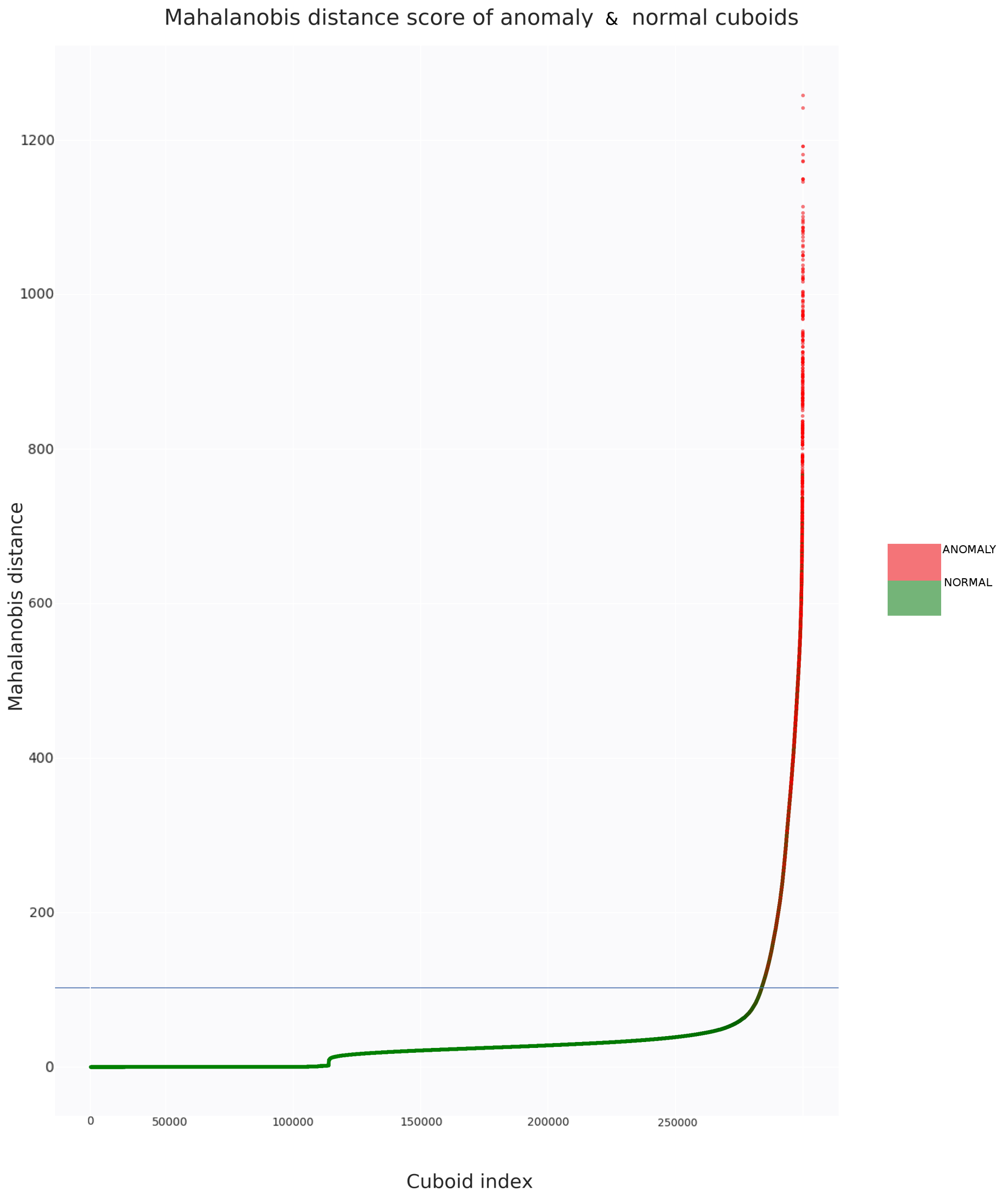


Figure 8–2 – Mahalanobis distances calculated from the validation-set on the artificial dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

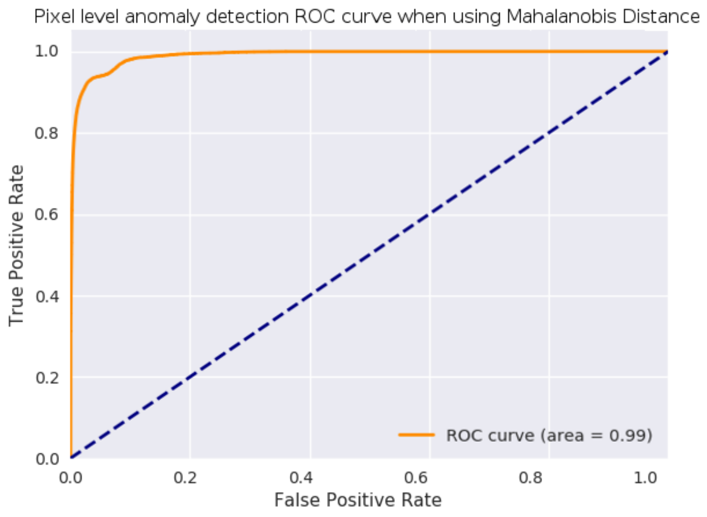


Figure 8–3 – ROC curve for the artificial dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

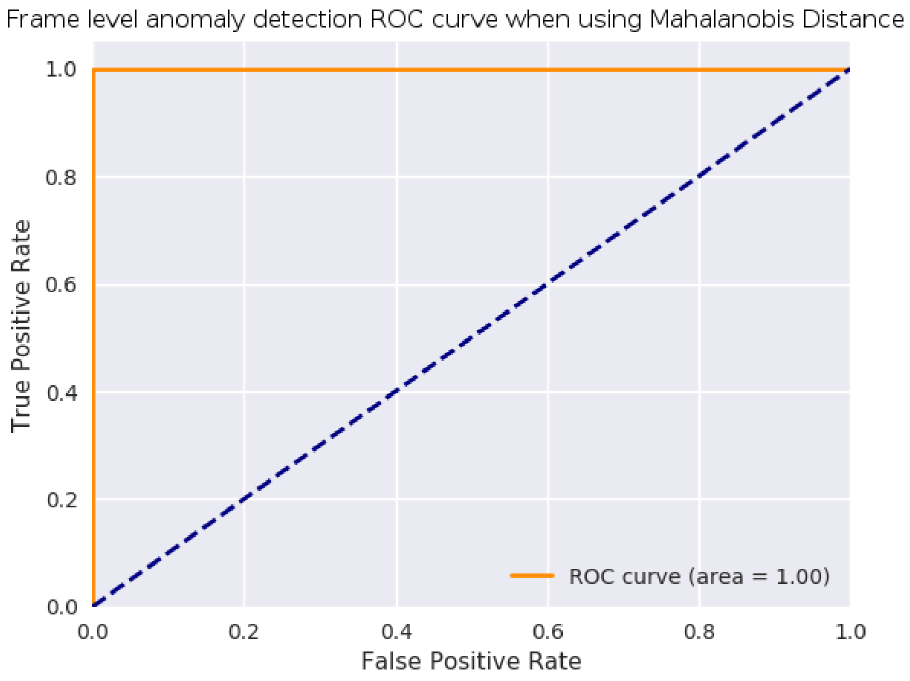


Figure 8–4 – ROC curve for the artificial dataset. Classification at the frame level. Dotted blue line represents a random classifier.

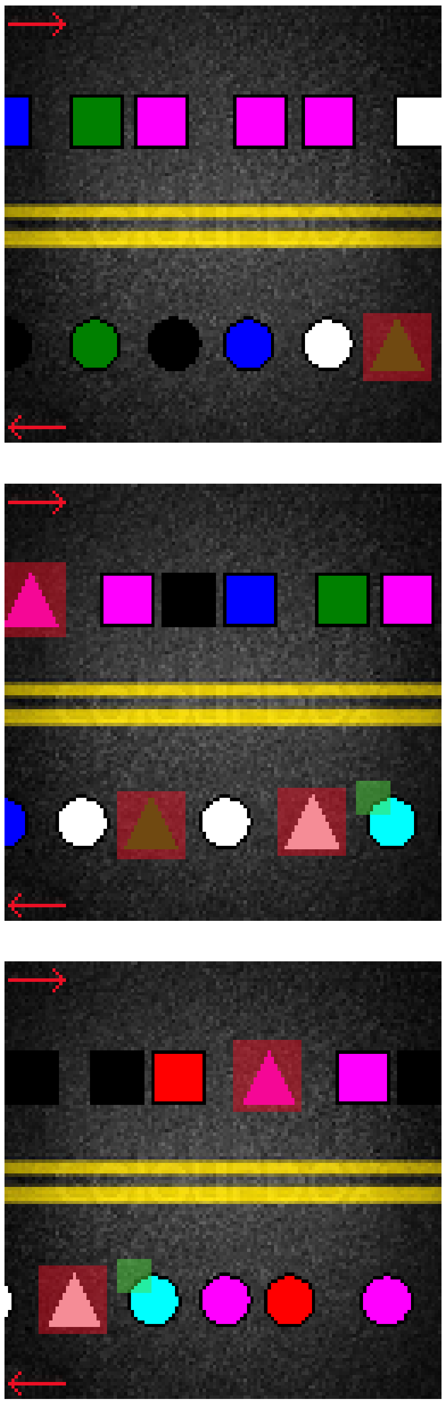


Figure 8–5 – Illustration of spatio-temporal cuboid classification on the artificial dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.3 Camouflage dataset

The results on the datasets that follow will be presented in the same template as above. The results from the Camouflage dataset are given in Table 8–3.

Training frames	20
Testing frames	1440
Anomalies	A person moving back and forth across the scene.
Distribution of Mahalanobis distances	Fig 8–6
ROC curve (pixel level)	Fig 8–7
ROC curve (frame level)	Fig 8–8
Maximum accuracy	100
Maximum F1-score	100
Examples of test-set images	Fig 8–9
Analysis of misclassifications	No misclassifications. This is because the training set contains only static frames without any motion. Any motion in the test set is labelled as an anomaly, and hence, the model is reduced to being a motion detector.
Notes	Using a deep learning model for this dataset is unnecessary.
Hyperparameters	Table 8–4

Table 8–3 – Results from the Camouflage dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	128
K (Number of Gaussians, Number of cluster centroids)	4
R (Number of epochs)	60
Threshold of Mahalanobis distance score for maximum F1 score	1.9
λ	0.001

Table 8–4 – Hyperparameters for evaluating the model on the Camouflage dataset.

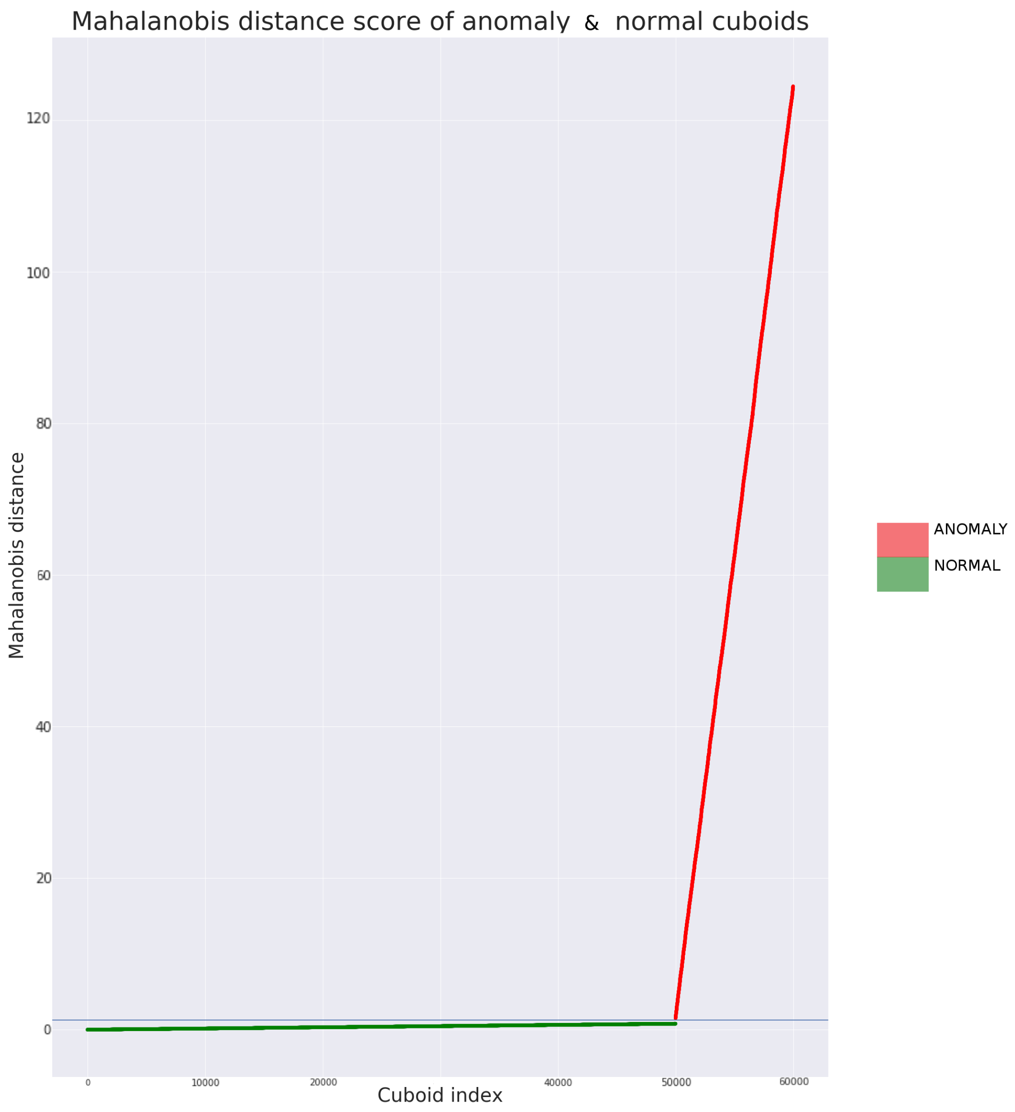


Figure 8–6 – Mahalanobis distances evaluated from the validation-set on the Camouflage dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

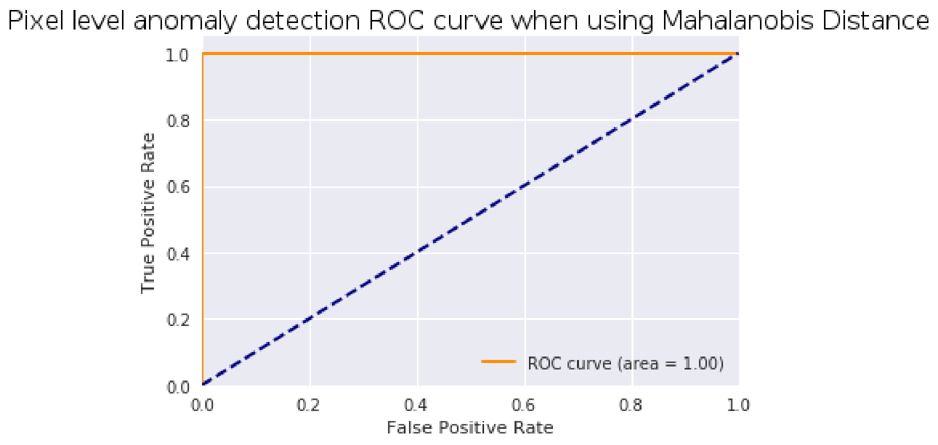


Figure 8–7 – ROC curve for the Camouflage dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

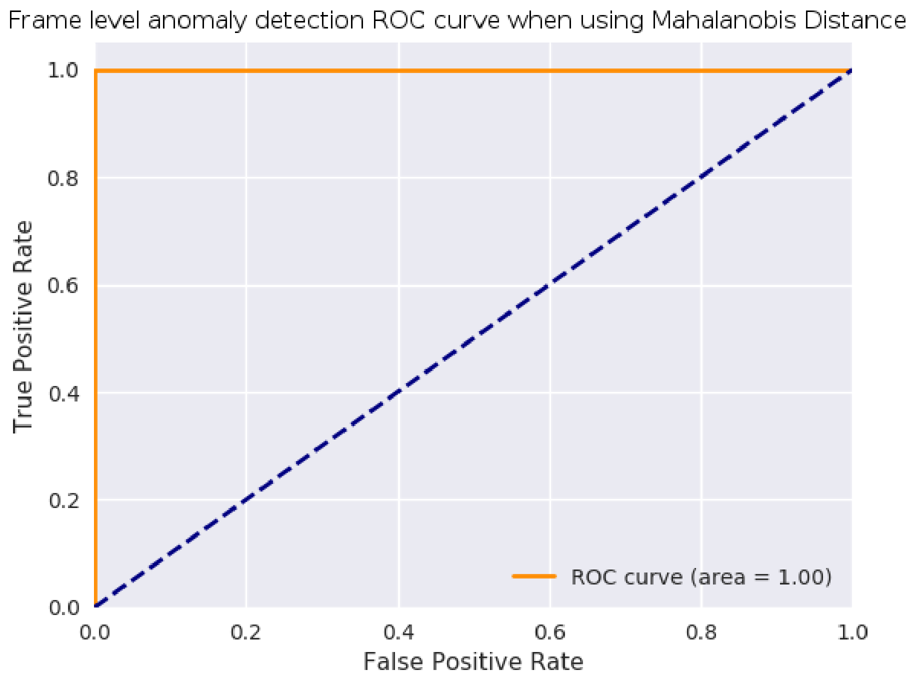


Figure 8–8 – ROC curve for the Camouflage dataset. Classification at the frame level. Dotted blue line represents a random classifier.

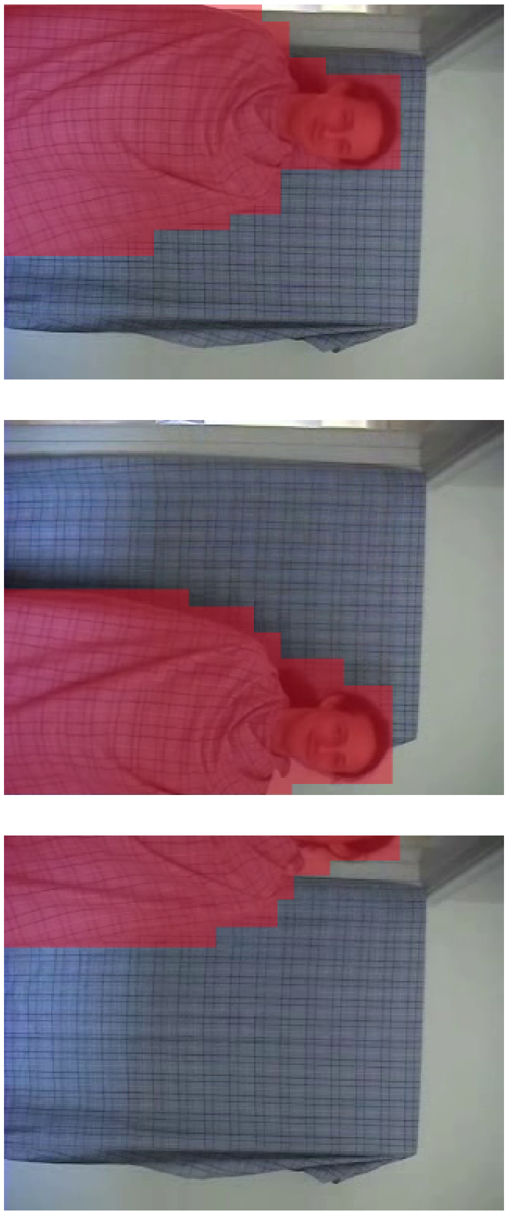


Figure 8–9 – Illustration of spatio-temporal cuboid classification on Camouflage dataset. Red hue is for correctly classified spatio-temporal cuboids. It may be recollected that the task at hand for the Camouflage dataset is to detect the moving person.

8.4 Boat-Sea dataset

Details on evaluating the model on the Boat-Sea dataset can be found in Table 8–5.

Training frames	200
Testing frames	250
Anomalies	A boat floating into the scene.
Distribution of Mahalanobis distances	Fig 8–10
ROC curve (pixel level)	Fig 8–11
ROC curve (frame level)	Fig 8–12
Maximum accuracy	94.03
Maximum F1-score	70.39
Examples of test-set images	Fig 8–13
Analysis of misclassifications	It may be observed that the misclassification of some spatio-temporal cuboids are because of the waves caused in the sea due to the boat.
Notes	The model can be made stronger for this scene by having a larger training dataset. This will enable the model to assimiliate all the dynamic possibilities of the scene.
Hyperparameters	Table 8–6

Table 8–5 – Results from the Boat-Sea dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	128
K (Number of Gaussians, Number of cluster centroids)	20
R (Number of epochs)	300
Threshold of Mahalanobis distance score for maximum F1 score	68.3
λ	0.01

Table 8–6 – Hyperparameters for evaluating the model on the Boat-Sea dataset.

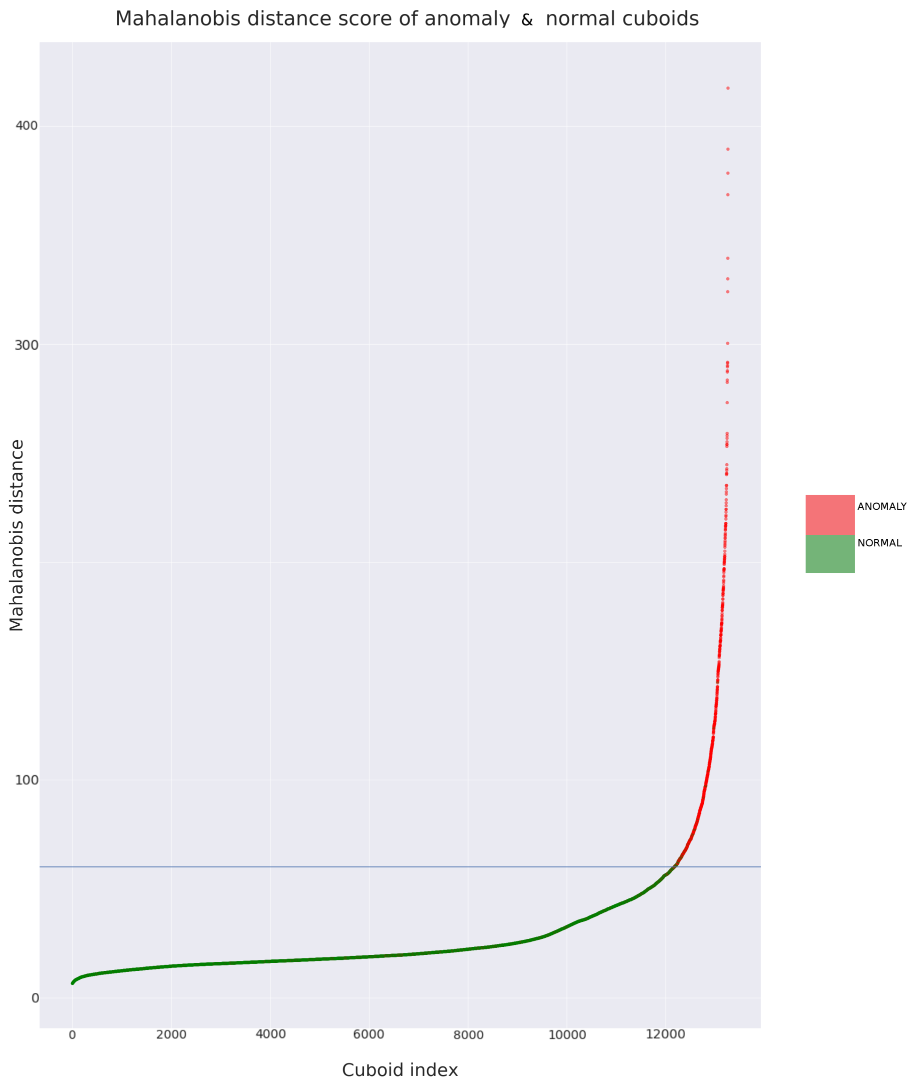


Figure 8–10 – Mahalanobis distances evaluated from the validation-set on the Boat-Sea dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

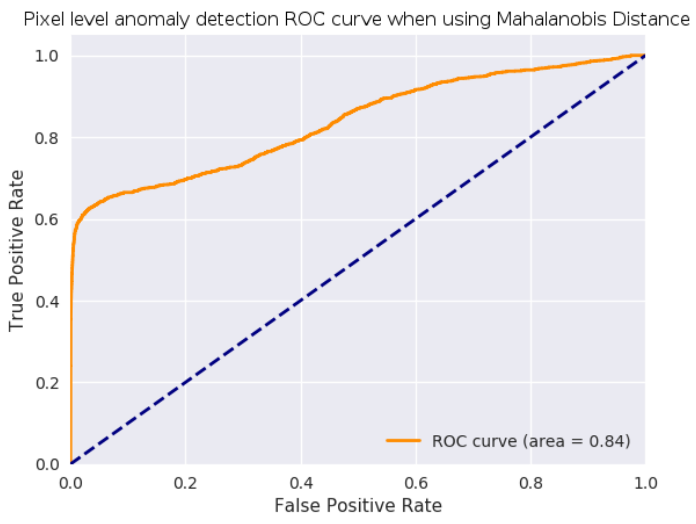


Figure 8–11 – ROC curve for the Boat-Sea dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

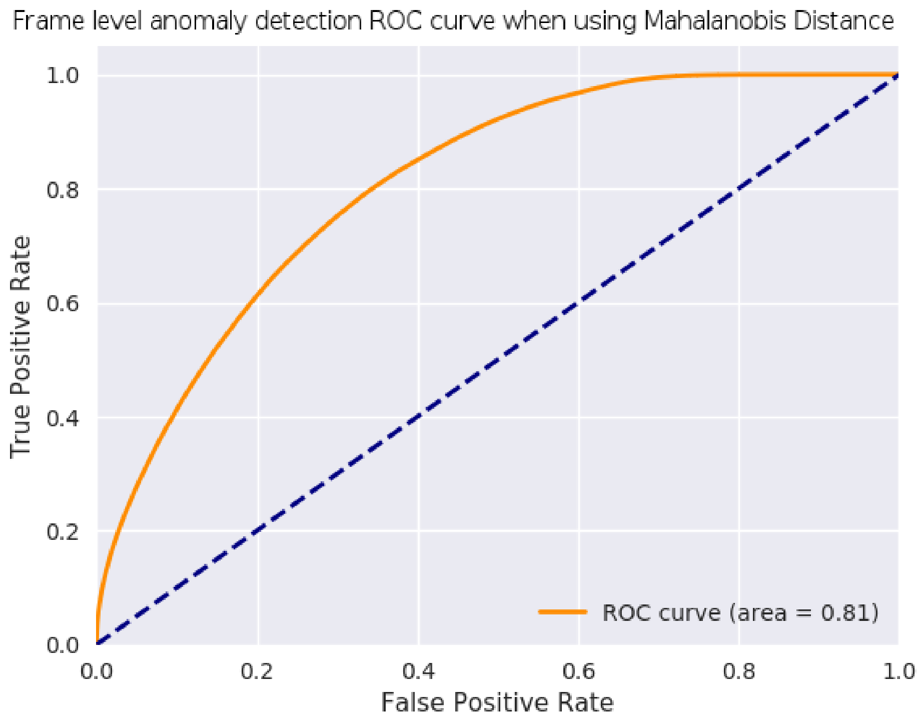


Figure 8–12 – ROC curve for the Boat-Sea dataset. Classification at the frame level. Dotted blue line represents a random classifier.

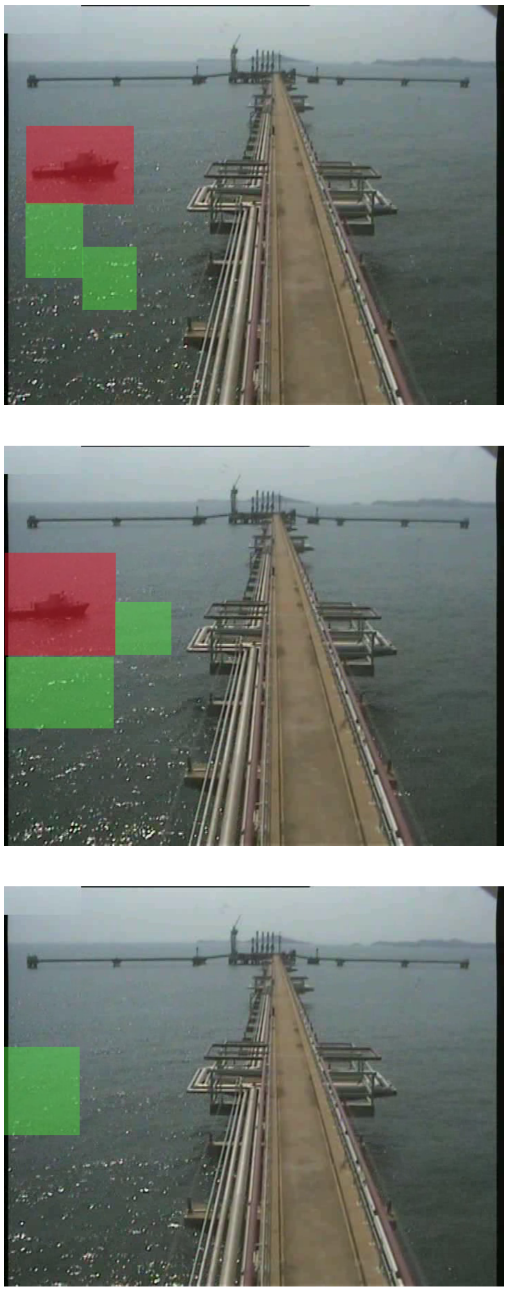


Figure 8–13 – Illustration of spatio-temporal cuboid classification on the Boat-Sea dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.5 Boat-River dataset

The details of model evaluation on the Boat-River dataset can be found in Table 8–7.

Training frames	100
Testing frames	235
Anomalies	A boat moving diagonally across the extent of the frame.
Distribution of Mahalanobis distances	Fig 8–14
ROC curve (pixel level)	Fig 8–15
ROC curve (frame level)	Fig 8–16
Maximum accuracy	87.4
Maximum F1-score	68.09
Examples of test-set images	Fig 8–17
Analysis of misclassifications	It is evident that the misclassifications are due to the wake of water caused by the speeding boat.
Notes	The dataset does not label the wake due to the boat as an anomaly. However, the model is observing a wake in the water for the first time. Hence it can be argued that with respect to the training set, the wake is indeed anomalous to this dataset. The boat is the cause and the wake is the effect. If the cause is an anomaly, why shouldn't the effect be? This is another evidence for the highly subjective nature of what is considered to be an anomaly.
Hyperparameters	Table 8–8

Table 8–7 – Results from the Boat-River dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	128
K (Number of Gaussians, Number of cluster centroids)	20
R (Number of epochs)	300
Threshold of Mahalanobis distance score for maximum F1 score	117.52
λ	0.01

Table 8–8 – Hyperparameters for evaluating the model on the Boat-River dataset.

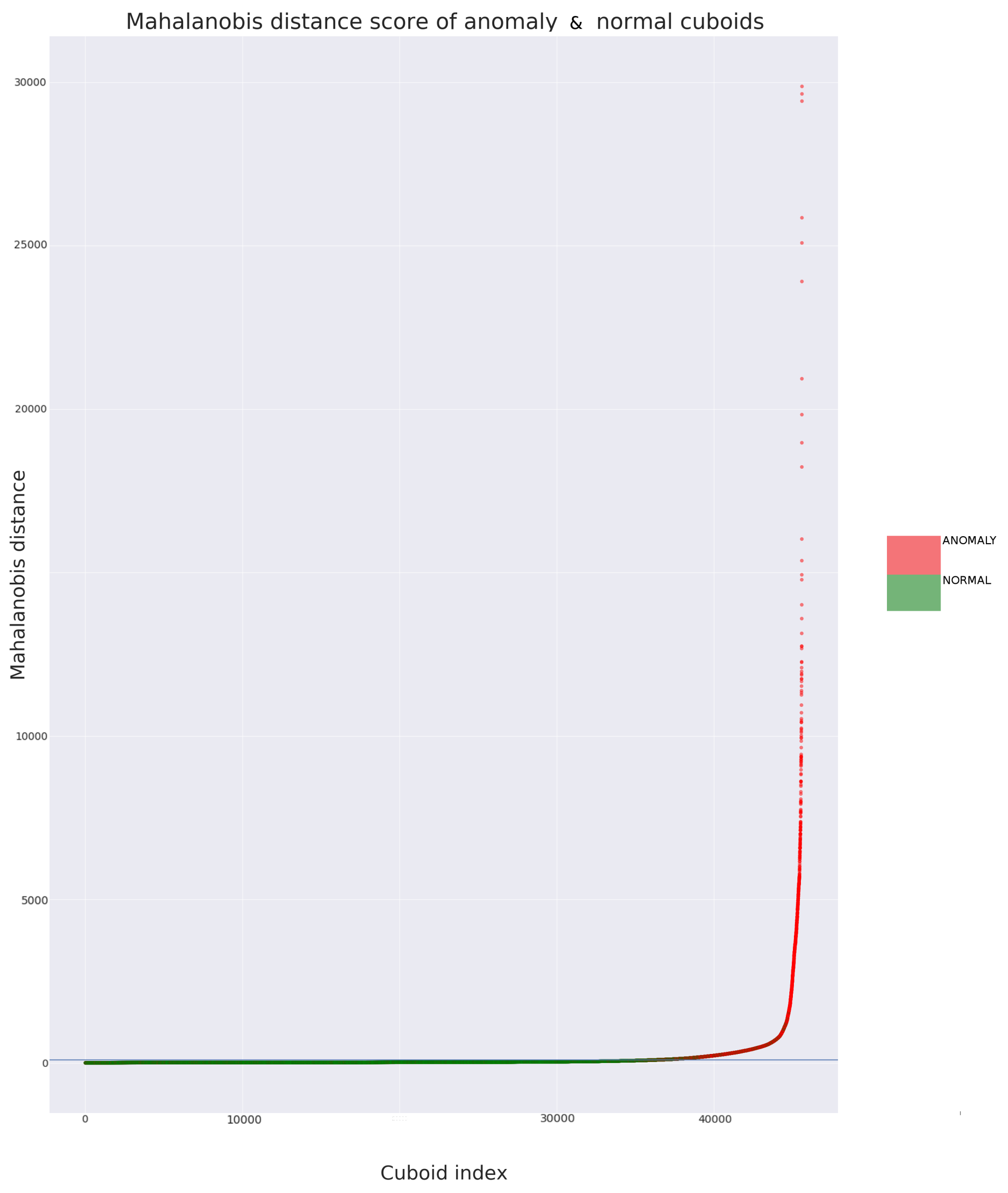


Figure 8–14 – Mahalanobis distances evaluated from the validation-set on the Boat-River dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

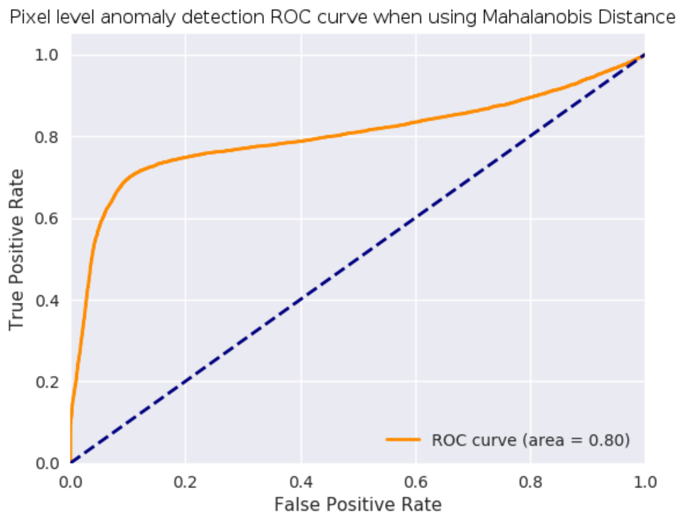


Figure 8–15 – ROC curve for the Boat-River dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

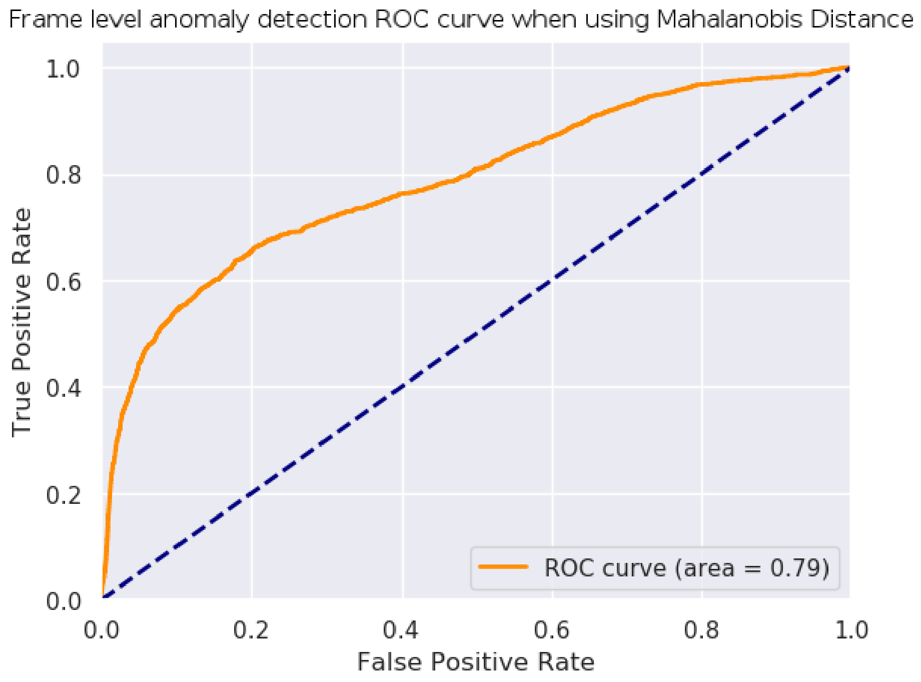


Figure 8–16 – ROC curve for the Boat-River dataset. Classification at the frame level. Dotted blue line represents a random classifier.

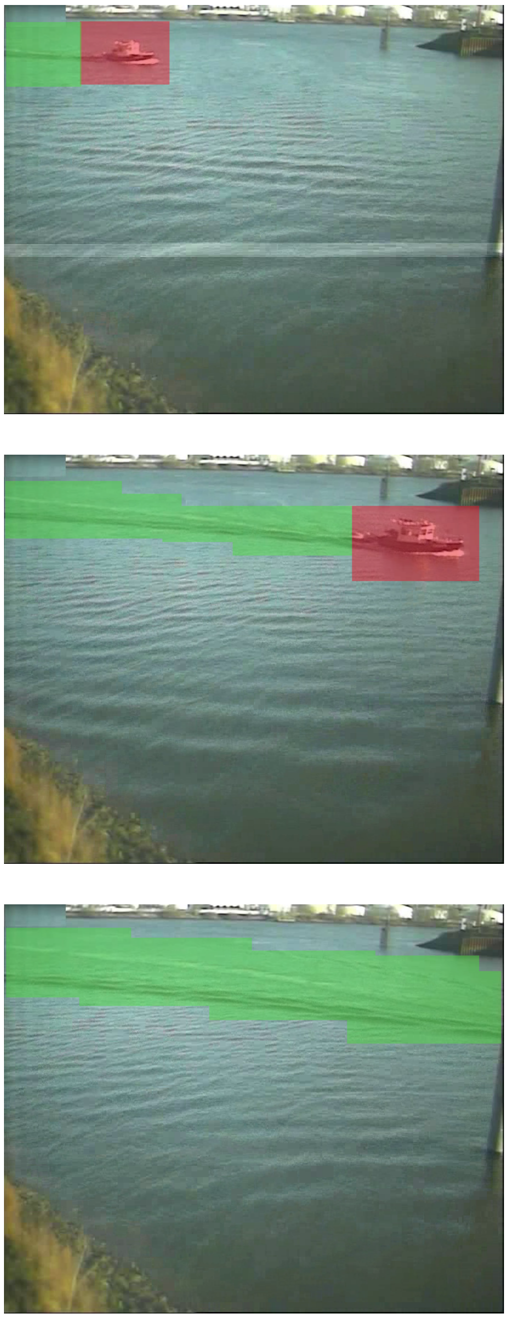


Figure 8–17 – Illustration of spatio-temporal cuboid classification on the Boat-River dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.6 Canoe dataset

The details of evaluating the model on the Canoe dataset can be found in Table 8–9.

Training frames	200
Testing frames	800
Anomalies	A canoe consisting of a family moving from left to right, across the frame.
Distribution of Mahalanobis distances	Fig 8–18
ROC curve (pixel level)	Fig 8–19
ROC curve (frame level)	Fig 8–20
Maximum accuracy	93
Maximum F1-score	86.3
Examples of test-set images	Fig 8–21
Analysis of misclassifications	It can be noticed that the incorrect classifications are either due to the wake of the boat on the river or the shadows of the passengers on the water.
Notes	Similarly to the Boat-River dataset, the agitation of water due to the canoe or the shadows of the passengers are not marked as anomalies in the evaluation set. However, in terms of the algorithm and the model, these are anomalies. Just like the canoe and its passengers, the model is encountering the wake and the shadows of its passengers for the first time. This is yet another example of the uncertainty and subjective nature of what is truly an anomaly.
Hyperparameters	Table 8–10

Table 8–9 – Results from the Canoe dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	32
K (Number of Gaussians, Number of cluster centroids)	10
R (Number of epochs)	80
Threshold of Mahalanobis distance score for maximum F1 score	82.5
λ	0.01

Table 8–10 – Hyperparameters for evaluating the model on the Canoe dataset.

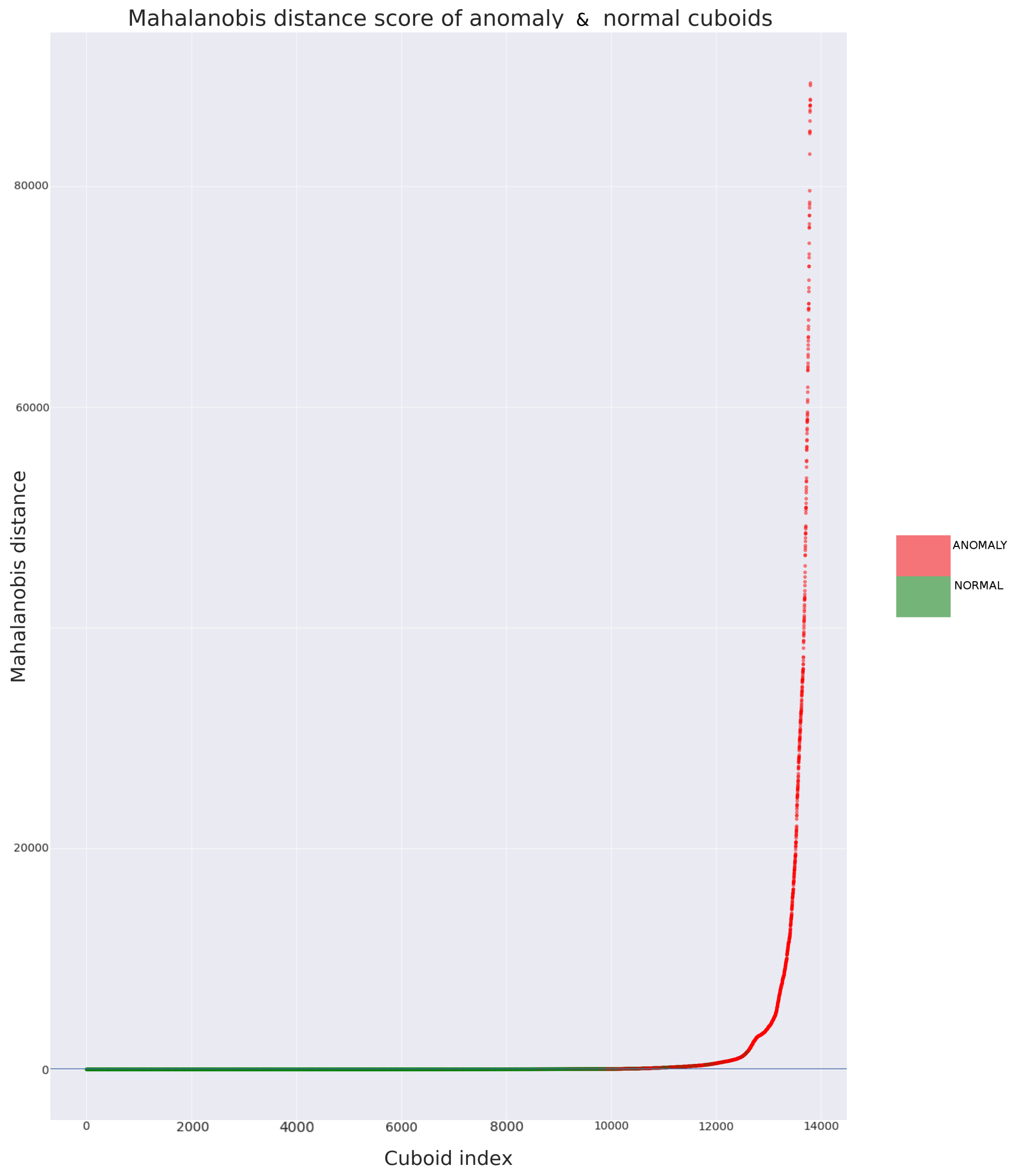


Figure 8–18 – Mahalanobis distances evaluated from the validation-set on the Canoe dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

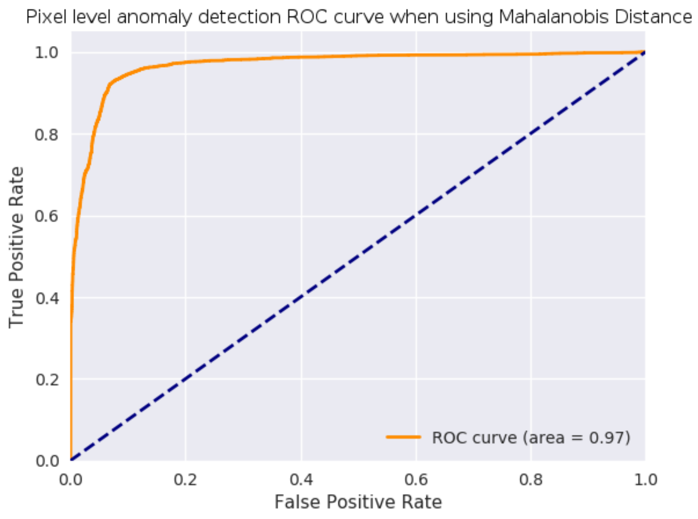


Figure 8–19 – ROC curve for the Canoe dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

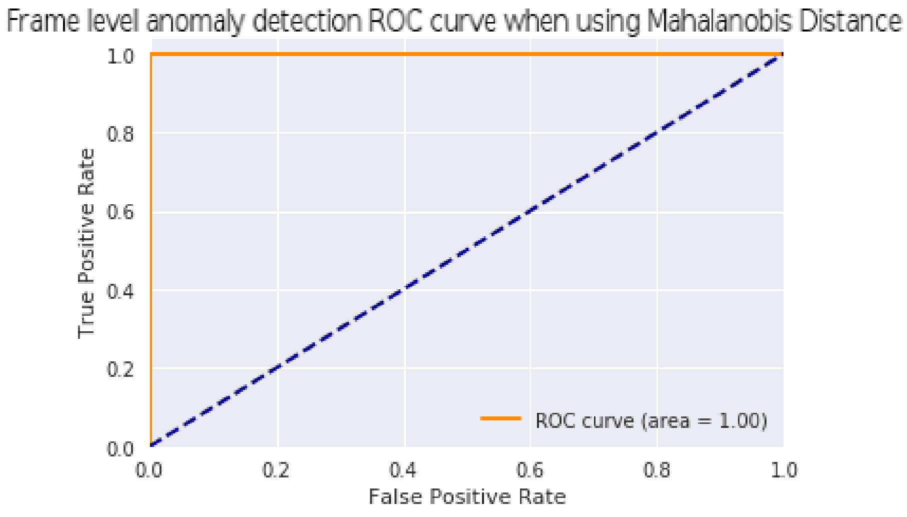


Figure 8–20 – ROC curve for the Canoe dataset. Classification at the frame level. Dotted blue line represents a random classifier.

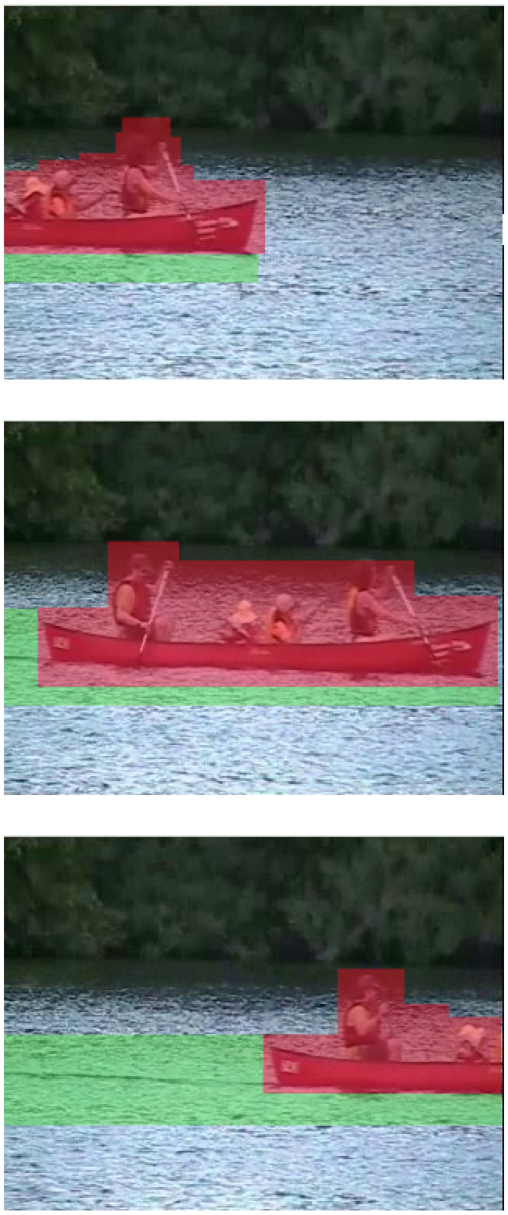


Figure 8–21 – Illustration of spatio-temporal cuboid classification on the Canoe dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.7 Train dataset

The details of evaluation of the model on the Train dataset can be found in Table 8–11.

Training frames	500
Testing frames	18000
Anomalies	Movement of people along the aisles of the train.
Distribution of Mahalanobis distances	Fig 8–22
ROC curve (pixel level)	Fig 8–23
ROC curve (frame level)	Fig 8–24
Maximum accuracy	86.8
Maximum F1-score	33.0
Examples of test-set images	Fig 8–25
Analysis of misclassifications	The misclassified regions are the windows (due to the view variations), the shadows of a few passengers, and the reflections of some passengers.
Notes	The performance of the model can probably be increased by extending the training set of the data by a lot more frames so that the model gets a chance to understand all the varying components of the scene. Experimental support for the above claim can be found in Section 9.2.
Hyperparameters	Table 8–12

Table 8–11 – Results from the Train dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	256
K (Number of Gaussians, Number of cluster centroids)	20
R (Number of epochs)	300
Threshold of Mahalanobis distance score for maximum F1 score	2989.86
λ	0.001

Table 8–12 – Hyperparameters for evaluating the model on the Train dataset.

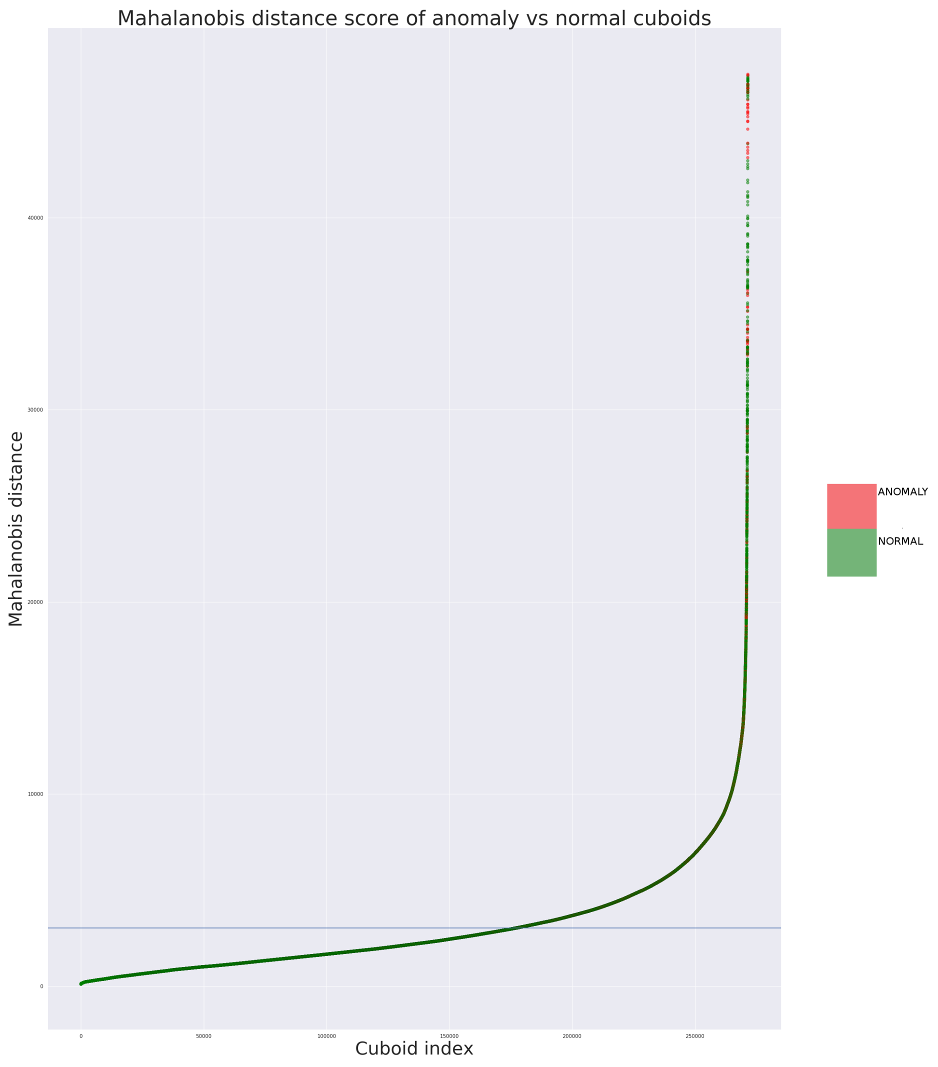


Figure 8–22 – Mahalanobis distances evaluated from the validation-set on the Train dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

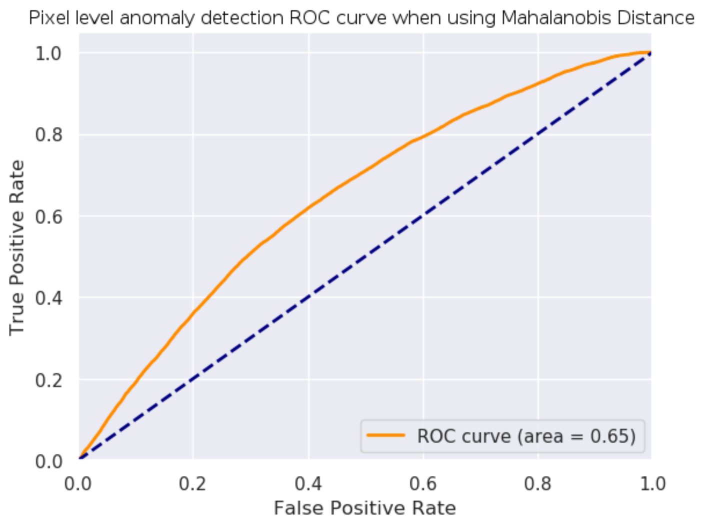


Figure 8–23 – ROC curve for the Train dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

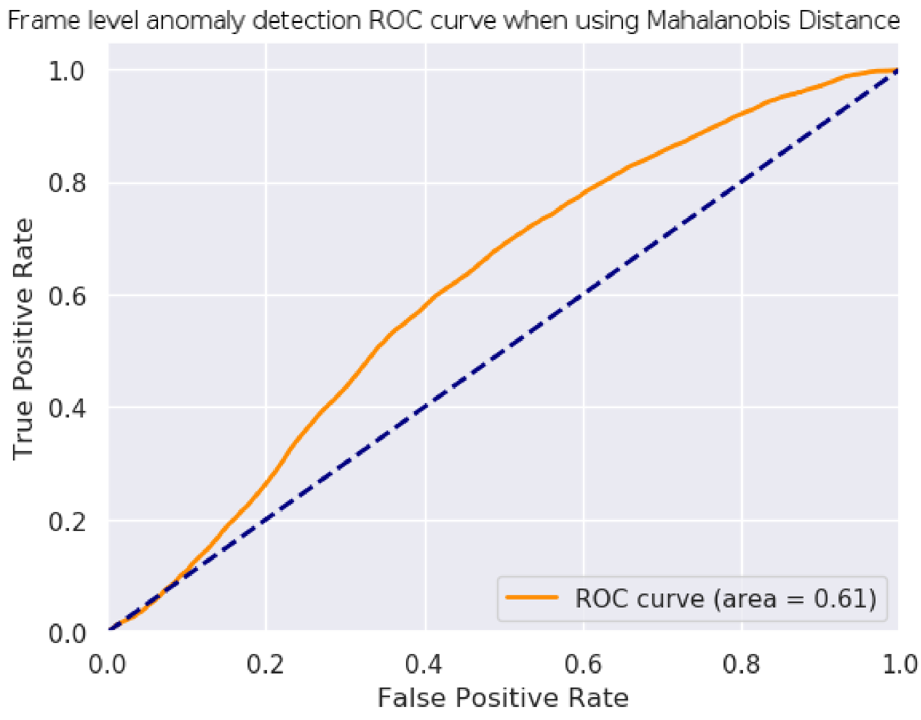


Figure 8–24 – ROC curve for the Train dataset. Classification at the frame level. Dotted blue line represents a random classifier.

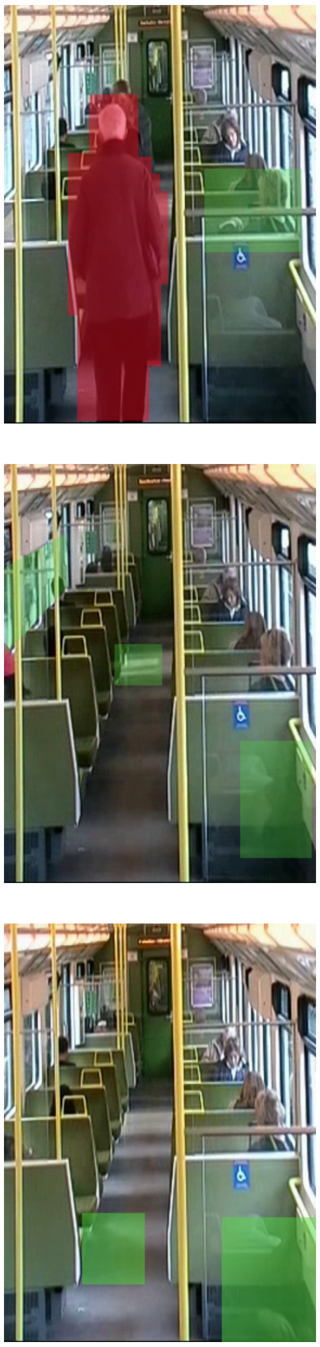


Figure 8–25 – Illustration of spatio-temporal cuboid classification on the Train dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.8 UCSD Ped 1 Dataset

The results from the UCSD Ped 1 dataset can be observed in Table 8–13.

Training frames	6800
Testing frames	7200
Anomalies	Bicycles, vehicles, skateboarders and people on wheelchairs.
Distribution of Mahalanobis distances	Fig 8–26
ROC curve (pixel level)	Fig 8–27
ROC curve (frame level)	Fig 8–28
Maximum accuracy	85.42
Maximum F1-score	37.21
Examples of test-set images	Fig 8–29
Analysis of misclassifications	Misclassifications include crowds and pedestrians moving very close to anomalous events like bicycles.
Notes	This is due to the fact that the misclassified instances are also very few in the training set. This dataset is another instance where an extensive training set would have helped the model identify the anomalies accurately.
Hyperparameters	Table 8–14

Table 8–13 – Results from the UCSD Ped 1 dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	32
K (Number of Gaussians, Number of cluster centroids)	20
R (Number of epochs)	110
Threshold of Mahalanobis distance score for maximum F1 score	24.47
λ	0.01

Table 8–14 – Hyperparameters for evaluating the model on the UCSD Ped 1 dataset.

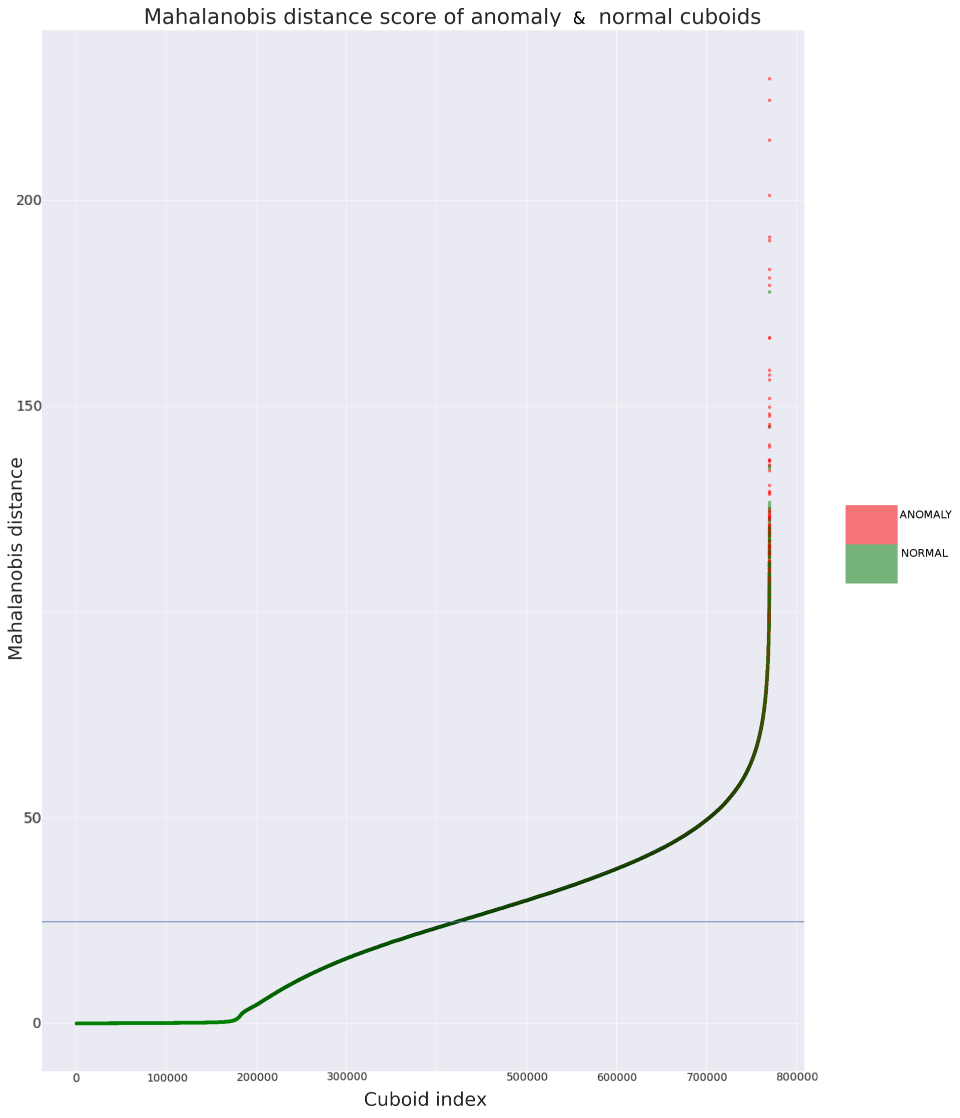


Figure 8–26 – Mahalanobis distances evaluated from the validation-set on the UCSD Ped 1 dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

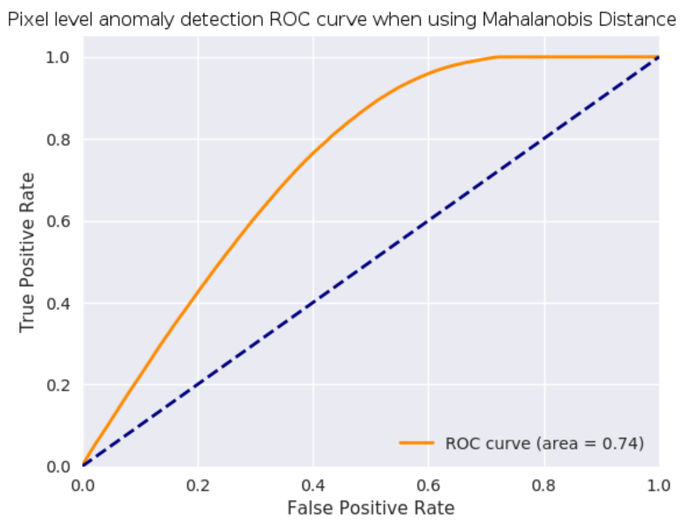


Figure 8–27 – ROC curve for the UCSD Ped 1 dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

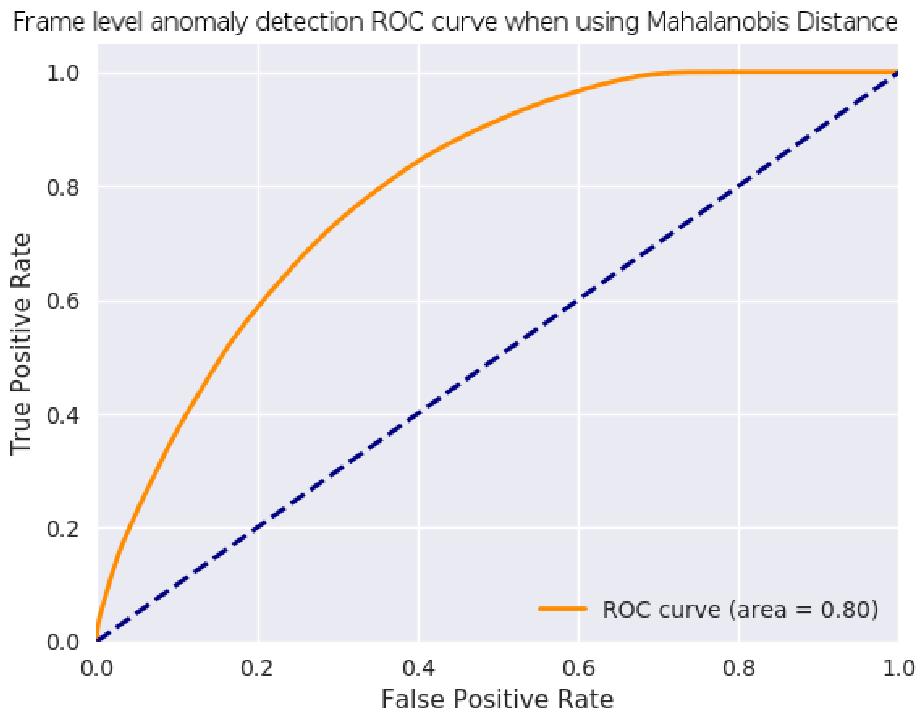


Figure 8–28 – ROC curve for the UCSD Ped 1 dataset. Classification at the frame level. Dotted blue line represents a random classifier.

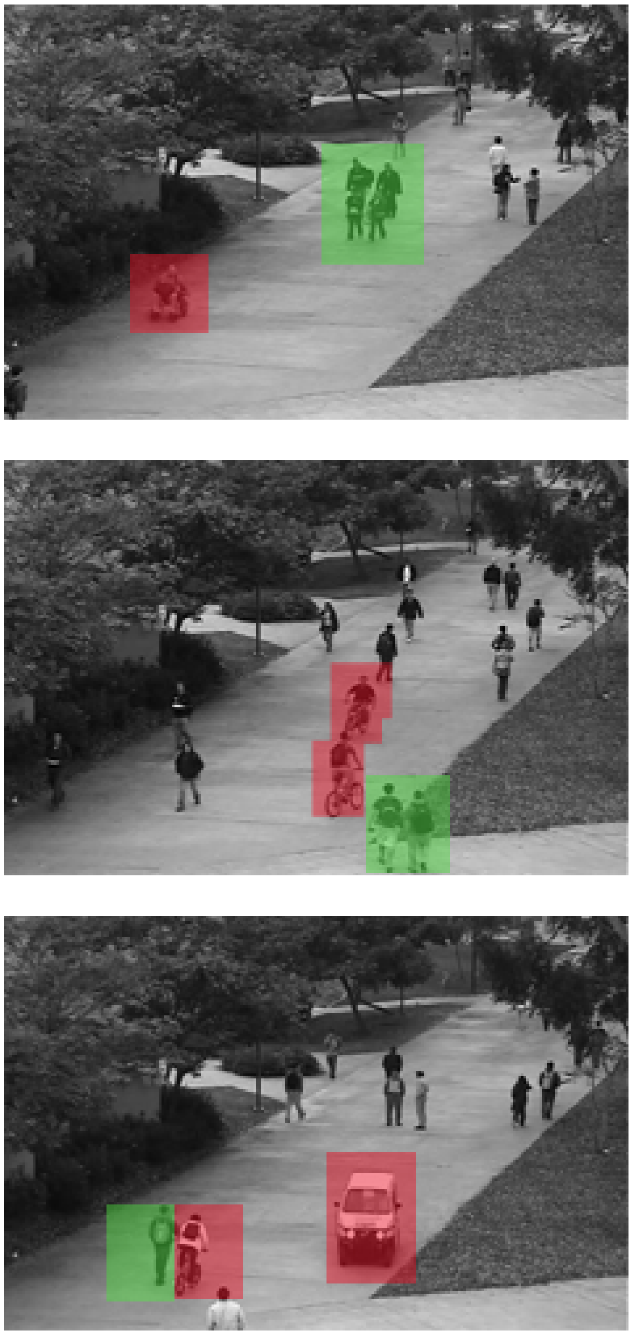


Figure 8–29 – Illustration of spatio-temporal cuboid classification on the UCSD Ped 1 dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.9 UCSD Ped 2 Dataset

The results from the UCSD Ped 2 dataset can be seen in Table 8–15.

Training frames	3200
Testing frames	2400
Anomalies	Bicycles, vehicles and skateboarders.
Distribution of Mahalanobis distances	Fig 8–30
ROC curve (pixel level)	Fig 8–31
ROC curve (frame level)	Fig 8–32
Maximum accuracy	88.03
Maximum F1-score	44.17
Examples of test-set images	Fig 8–33
Analysis of misclassifications	Similarly to UCSD Ped 1, the misclassifications include crowds and pedestrians very close to anomalous events like bicycles. Some pedestrians in less traversed areas are also wrongly misclassified as anomalies.
Notes	This is due to the fact that the misclassified instances are also very few in the training set. Like UCSD Ped 1, this is also an instance where an extensive training set would have helped the model identify the anomalies accurately.
Hyperparameters	Table 8–16

Table 8–15 – Results from the UCSD Ped 2 dataset.

Hyperparameter	**Value**
n (Number of features in the encoding space)	32
K (Number of Gaussians, Number of cluster centroids)	20
R (Number of epochs)	75
Threshold of Mahalanobis distance score for maximum F1 score	22.94
λ	0.01

Table 8–16 – Hyperparameters for evaluating the model on the UCSD Ped 2 dataset.

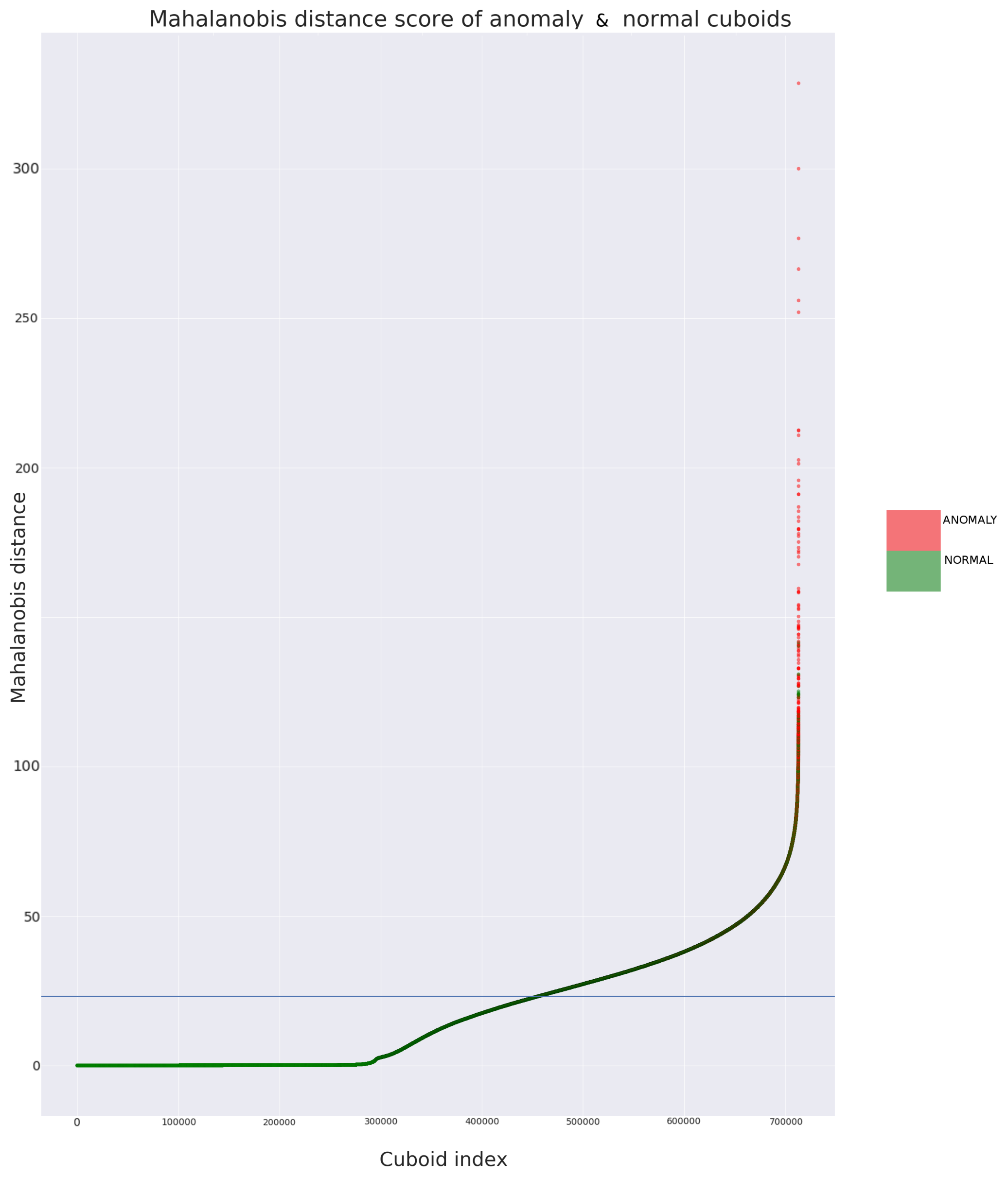


Figure 8–30 – Mahalanobis distances evaluated from the validation-set on the UCSD Ped 2 dataset. The threshold level for the maximum F1 score from the distribution is depicted in blue.

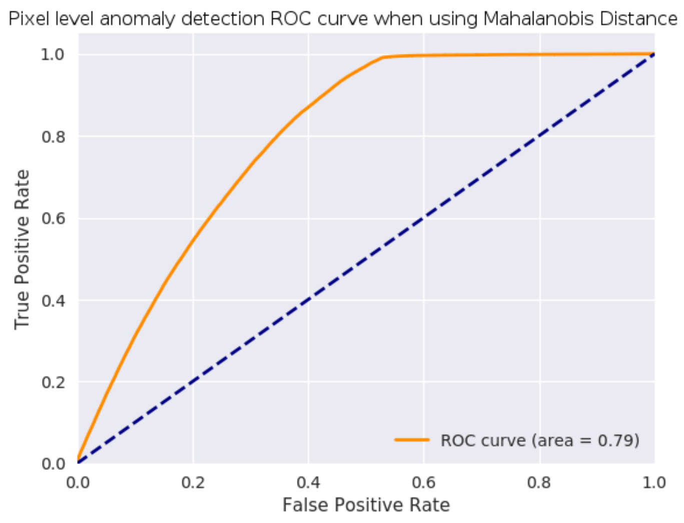


Figure 8–31 – ROC curve for the UCSD Ped 2 dataset. Classification at the spatio-temporal cuboid level. Dotted blue line represents a random classifier.

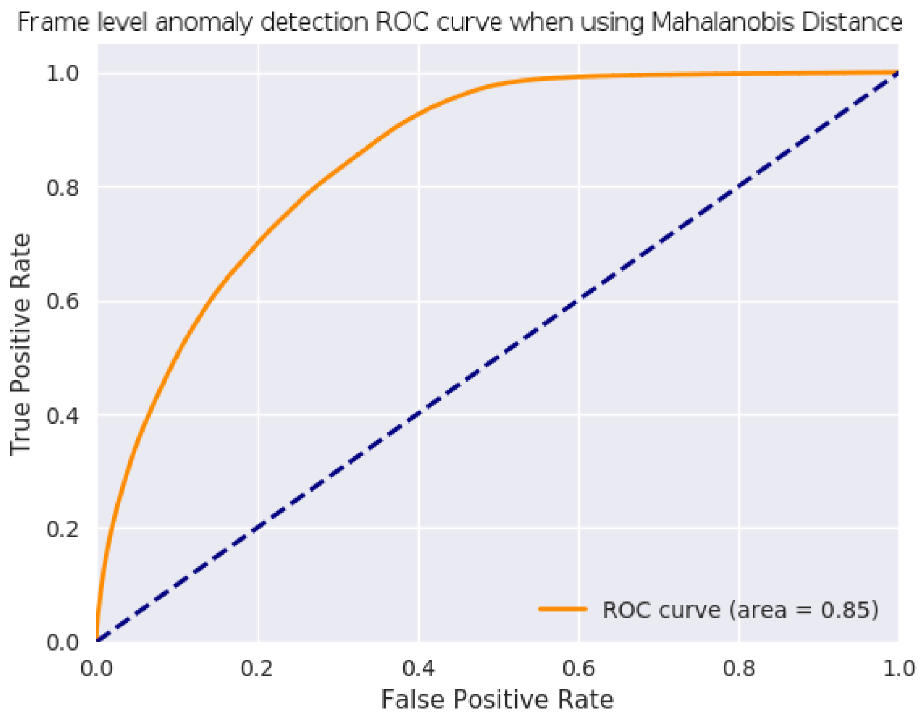


Figure 8–32 – ROC curve for the UCSD Ped 2 dataset. Classification at the frame level. Dotted blue line represents a random classifier.

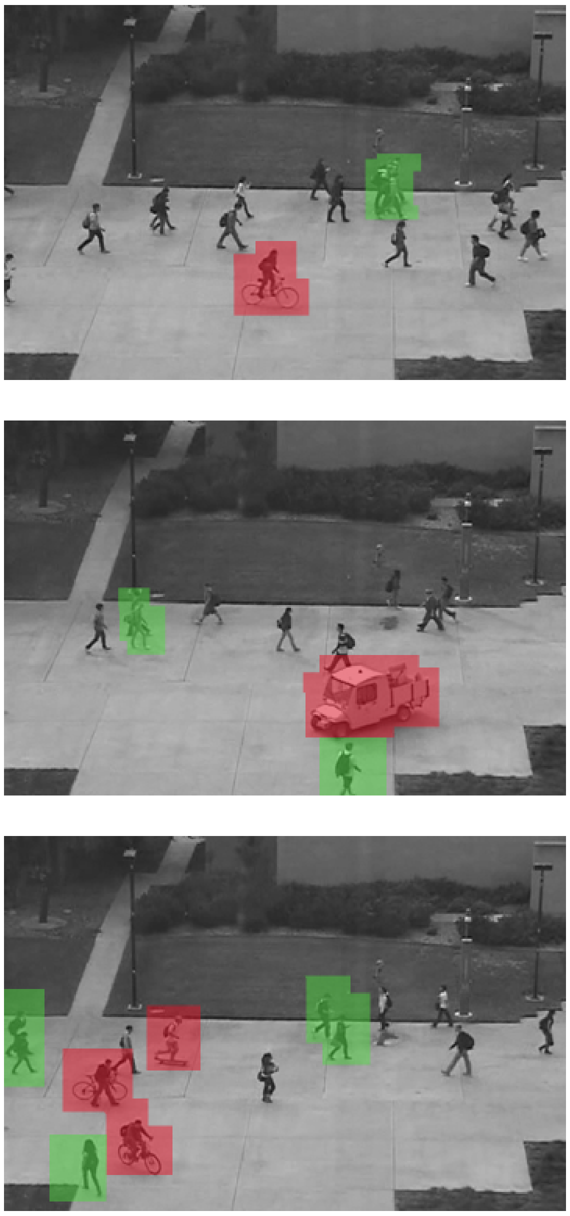


Figure 8–33 – Illustration of spatio-temporal cuboid classification on the UCSD Ped 2 dataset. (Score distribution thresholded to maximize the F1-score). Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids.

8.10 Comparison with the state-of-the-art

Most of the literature dealing with anomaly detection from videos, compare the performance of each algorithm on the basis of their results on the UCSD dataset. Consequently, we compare our algorithm with some top candidates for the state of the art in the UCSD datasets.

In Fig 8–34, ROC curves of our model are plotted along with the ROC curves from the social force model [46], the mixture of dynamic textures model [41], sparse reconstruction [14], fast detection at 150 fps model [40], statistical hypothesis detector [79], AMDN (Appearance and motion deep-net) [77] and GM (Gaussian mixture variational autoencoder) [19]. These results are from the UCSD Ped 1 dataset.

Similarly, in Fig 8–35, ROC curves from our algorithm can be observed along with the same from the social force model [46], mixture of dynamic textures model [41], deep-anomaly [61] and GM (Gaussian mixture variational autoencoder) [19]. These results are from the UCSD Ped 2 dataset.

From both figures, and with the assistance of Tables 8–17 and 8–18 it may be concluded that the results from our model can be considered competitive with all the other methods. It also has to be acknowledged that GM [19] is much better than our model in the frame-level evaluation since GM is a fusion model that treats the static training frames separately from the temporal content of the data, only to combine the information later in the model.

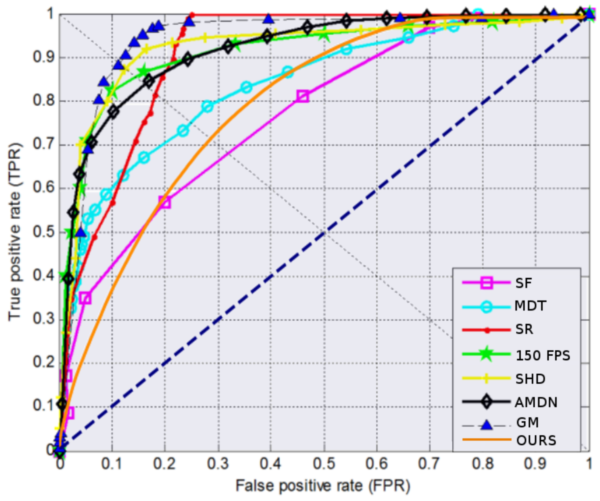
The slightly lower results from our model in the frame-level can be attributed to the fact that we are sampling our data in a spatio-temporal fashion. This means that some frames are still classified as anomalous after the cause of the anomaly has disappeared from the frame (but they exist in the spatio-temporal cuboids that sampled the event).

Dataset	**Best pixel level AUC in literature**	**Pixel level AUC in our model**
UCSD Ped 1	0.731 (SHD)	0.74
UCSD Ped 2	0.782 (GM)	0.79

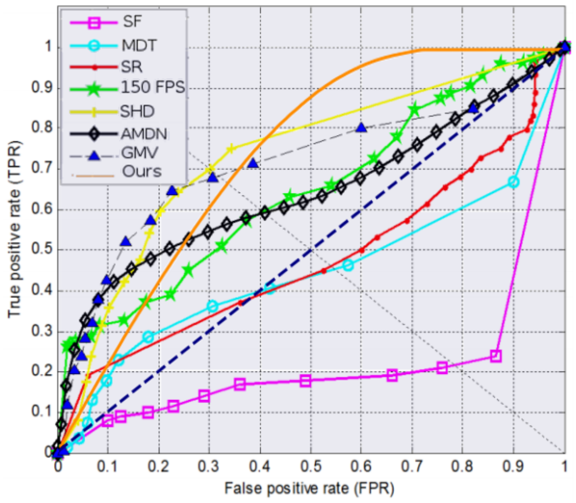
Table 8–17 – Comparison of our algorithm vs the best pixel level AUC (Area under ROC curve) in literature.

Dataset	**Best frame Level AUC in Literature**	**Frame level AUC in our model**
UCSD Ped 1	0.95 (GM)	0.80
UCSD Ped 2	0.922 (GM)	0.85

Table 8–18 – Comparison of our algorithm vs the best frame level AUC (Area under ROC curve) in literature.

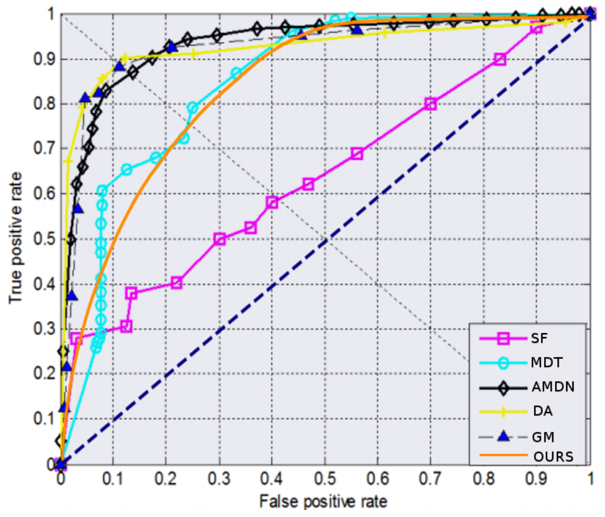


(1) Comparison of our algorithm with the state of the art algorithms in the frame level detection rate. USCD1 dataset.

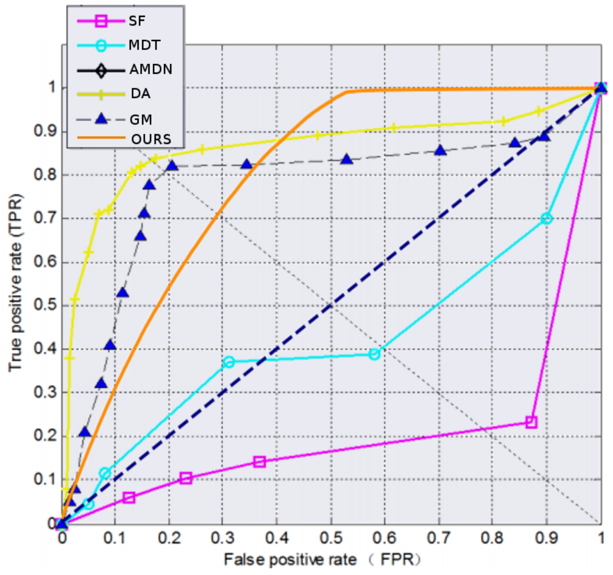


(2) Comparison of our algorithm with the state of the art algorithms in the pixel level detection rate. USCD1 dataset.

Figure 8–34 – Comparison of the ROC curves of our algorithm vs SF [46], MDT [41], SR [14], Detection at 150 FPS [40], SHD [79] , AMDN [77]. and GM [19]. Results from UCSD Ped 1 dataset.



(1) Comparison of our algorithm with the state of the art algorithms in the frame level detection rate. UCSD Ped 2 dataset.



(2) Comparison of our algorithm with the state of the art algorithms in the pixel level detection rate. UCSD Ped 2 dataset.

Figure 8–35 – Comparison of the ROC curves of our algorithm vs SF [46], MDT [41], DA [61], GM [19]. Results from the UCSD Ped 2 dataset.

Chapter 9
Conclusion

In this chapter, we will present the important issues and lessons understood from this research work.

9.1 The proposed solution

This thesis presented an innovative solution for the detection and localization of anomalous events in a video stream from a static camera. It is a novel algorithm, previously not examined for this problem. Hence, this is a considerable addition to the field of research in anomaly detection from videos, especially since it has been demonstrated to be competitive with the state-of-the-art methods. The algorithm is general in nature and can be deployed to any scene as is. Additionally, it is very easy to implement in simple programming languages like Python [56] using software packages like Keras [11], Tensorflow [2] and Scikit-Learn [52].

The algorithm depends on some hyper-parameters that require tuning for its successful employment (λ, K, n and threshold of anomaly score). Since we are using a deep learning model, the training set for the model must be large enough to suitably train the autoencoder to convergence. However, despite these drawbacks, the proposed solution has proved to be a compelling candidate for performing anomaly detection from videos.

9.2 How important are the data?

In parallel to evaluating the proposed solution, we also studied the importance of having enough training data for the convolutional auto-encoder. In Fig 9–1, the AUC (Area Under ROC Curve) score of the best classifiers obtained in each dataset is plotted against the ratio of training frames to the test frames. However, all these datasets are not equally challenging. In the Canoe, Camouflage, Boat-River and Boat-Sea datasets, the classifier is used to detect an object previously non-existent in a fairly static scene. On the other hand, the Train, UCSD Ped 1, UCSD Ped 2 and Artificial datasets are more challenging and features more dynamic scenes from which anomalies need to be detected.

Fig 9–1 can be re-drawn by only preserving the data from the challenging datasets. Fig 9–2 depicts such a graph. A clearly evident trend can be observed in this plot, where the AUC score is monotonically increasing with an increase in the ratio of the number of training frames to the number of testing frames.

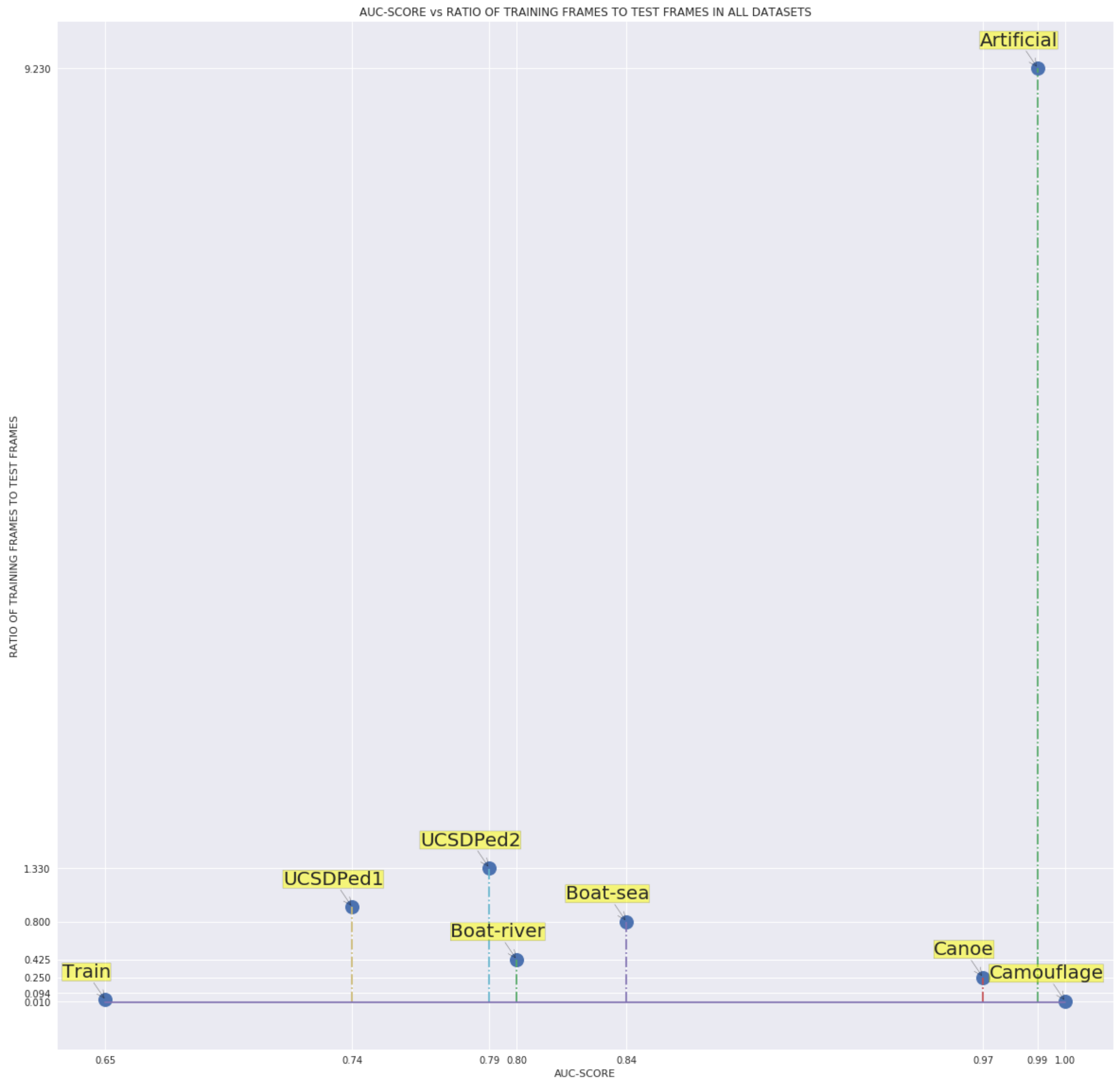


Figure 9–1 – The AUC score of the best classifiers for each dataset vs the ratio of training-frames to testing-frames.

Since we also have access to a system from which "unlimited" training data can be generated, we can use this to our advantage and generate training-sets of variable sizes. We can use these training-sets to re-examine this apparent relationship between the ratio of training and testing frames to the performance of the classifier. We ran an experiment by training the model on training-sets of variable sizes from the artificial dataset, while keeping the length of the test-set and hyper-parameters fixed. We used $\lambda = 0.01$, $K = 10$ and $n = 32$ for all the models. The results of this exercise can be observed in Fig 9–3. This verifies our understanding that more training data leads to better performing models.

It is well-known that when more training data are available, we can obtain better models. This fact is specifically true for auto-encoders because they aim to map the input distribution to a low-dimensional manifold and reconstruct it at the output. When the amount of available training data approaches "infinity" (in truth, a very large number), the auto-encoder is able to learn the true distribution of the data. In other words, an "infinite" amount of training data nullifies the training data bias.

Usually when models are trained on a dataset, we are concerned whether the size of the dataset is big enough to represent the true distribution effectively. Hence, the criticism (if any) of the model is most-likely biased towards not having sufficient data. Having a source for datasets of variable length give us the ability to study the true merits or limits *of the model itself.* The model can be analyzed and critiqued without the worry of data limitations.

Nearly all deep learning models have millions of parameters that require optimization. This means that a tremendous amount of training data are required for the models. Looking back at Table 3–1, we see how limited the training data from the benchmark datasets actually are. They have a very low number of training frames when compared to the number of parameters in a model. This characteristic of the datasets forbids many deep learning algorithms from being applied to the problem. Having a resource for "unlimited" data alleviates all of these concerns and opens the field of play to any deep parameter-heavy model.

Collecting a large, ground-truth annotated video dataset of a natural scene is indeed exasperating and often impossible. The UCSD dataset (used in this research) was released in 2008. A larger, more dynamic dataset of real videos, specifically aimed at deep learning has not yet been developed. This has been a goal for this research. Our framework for limitless ground-truth annotated video data is aimed at deep learning models. The user can control the randomness, the activity rate and the length of the videos. This is a first of its kind for anomaly detection from videos. Albeit simple, it is a very important stepping stone towards developing very good models for anomaly detection from videos.

The difference between fundamental shapes (squares, circles and triangles) are more evident than the difference between a person holding a coffee cup and a person weilding a gun. So, if an anomaly detection model cannot detect triangles as anomalies from a dataset that usually contains only squares and circles, it will most likely be unable to differentiate a person holding a gun and a person holding a coffee cup. In short, if a model does not perform well on the artificial data, it may be expected not to work well for real data.

The converse may also be true. A model might perform extremely well on simulated data, but may fail for real data. However, developing more realistic video simulations for anomaly detection using computer graphics engines could help in establishing concrete conlusions for this problem.

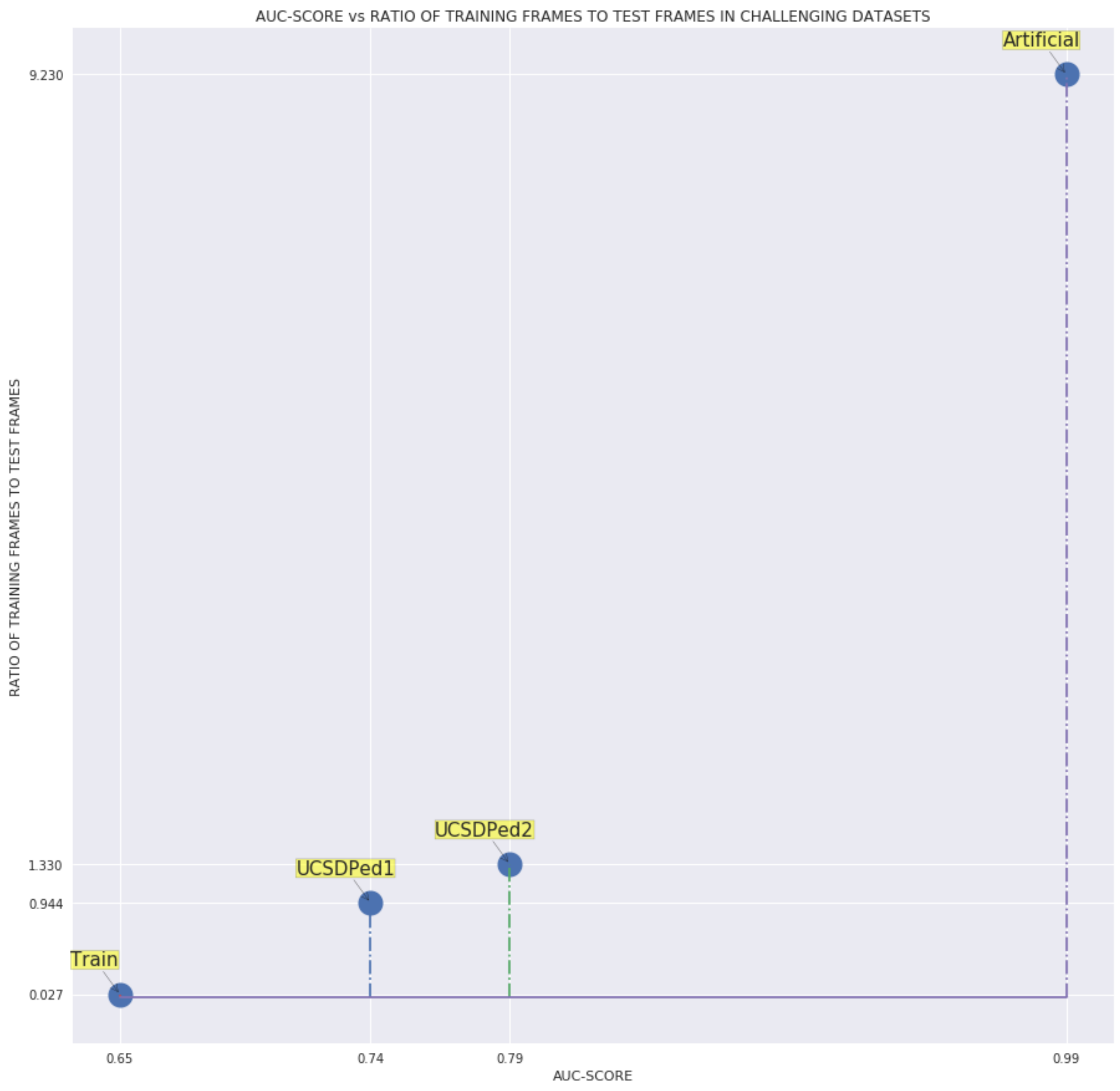


Figure 9–2 – The AUC score of the best classifiers for each challenging dataset vs the ratio of training-frames to testing-frames.

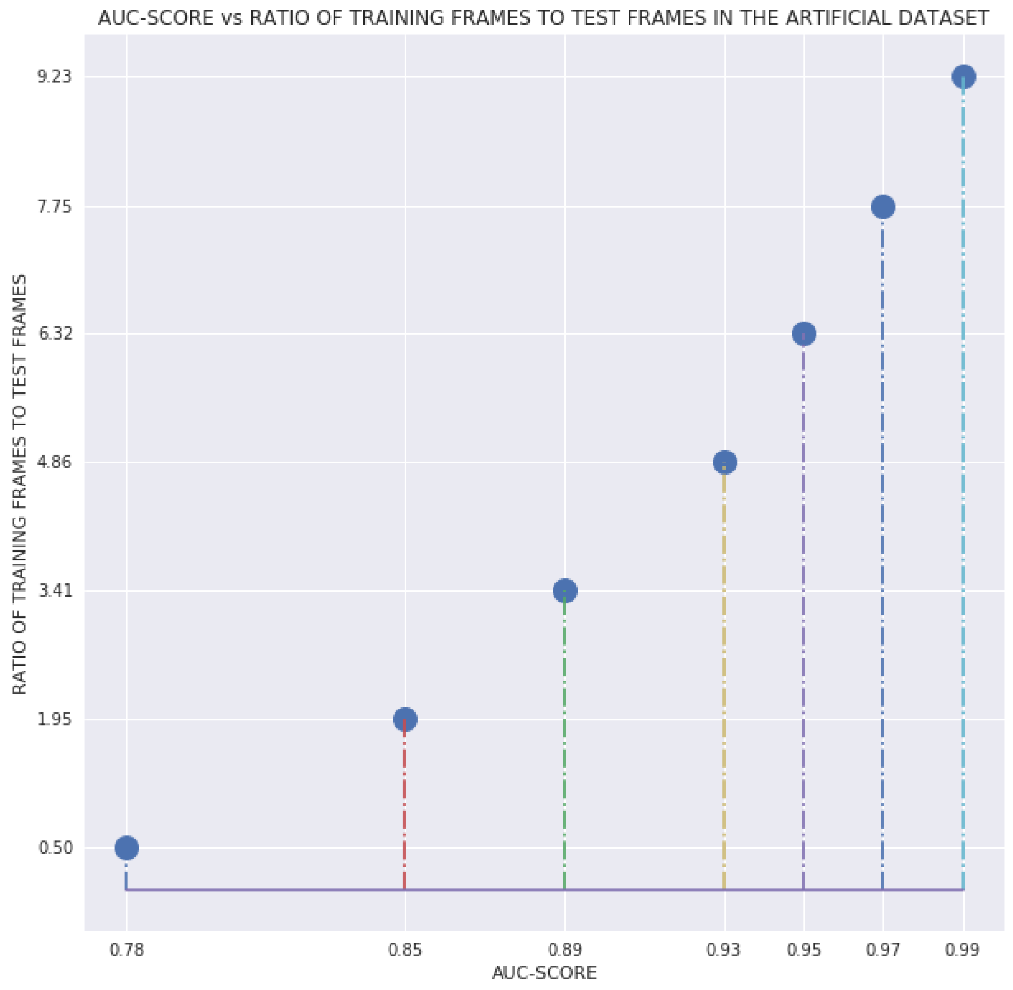


Figure 9–3 – The AUC scores of the classifier for varying training-set sizes of the artificial dataset. AUC scores on the X-axis and the ratio of the training-frames to the testing frames on the Y-axis.

9.3 What is an anomaly, really?

This is really a philosophical question. The proverb "one man's food is another man's poison" comes to mind as a summary when considering the subjective nature of an anomaly.

Let us take the Canoe dataset as a simple example to do some thought experiments. Fig 8–21 shows some examples of the results when using our model on the Canoe dataset. The dataset[1] has labelled only the canoe and its passengers as anomalies. However, when the canoe drifts into the scene, the reflection of the passengers can be seen in the water. When the canoe passes, there is a wake in the water following the canoe. If the canoe and its passengers (the cause) have been marked as anomalies, why is the wake or the reflection of the passengers (effect) not marked as anomalies? The effect follows the cause. If the cause is an anomaly, in our opinion, the effect must also be marked as anomalous. From a practical point of view, we think that it is better to involve a human arbiter for anything even remotely suspicious than to regret the choice of skipping an event in hindsight.

As an example of this subjective nature of an anomaly, we conducted some experiments. The Boat-Sea (Fig 3–3) and Boat-River (Fig 3–5) datasets are quite similar. Both datasets require the algorithm to detect a boat in an otherwise empty waterbody. We plotted the thresholds of the Mahalanobis distance score that gives us the maximum F1 score in both these datasets by varying K (the number of cluster centroids, Gaussians). All the other hyper-parameters were kept constant ($n = 32$, $\lambda = 0.001$). The result of this experiment can be observed in Fig 9–4. It can be noted how different the thresholds (or, the optimum cut-off anomaly score) for these datasets are. They are not just different, but even on a different scale.

Additionally, we can compare the thresholds obtained by running the expectation maximization algorithm for fitting the Gaussian mixture model on the same trained encoder ($E(\gamma)$) multiple times. Upon observing Fig 9–6 and Fig 9–5, it is also evident how stochastic

[1] The person/group who annotated the dataset.

the thresholds can be. Apart from being just subjective, the threshold for determining what truly might be an anomaly is also indefinite. This demonstrates how difficult it is to truly "establish" what an anomaly is. It is uncertain not only in the domain of definitions and human-interpretation of videos, but also in the feature-space domain.

This uncertainty and the stochastic nature of the data explains why the results from our model on the benchmark datasets have a considerable variance.

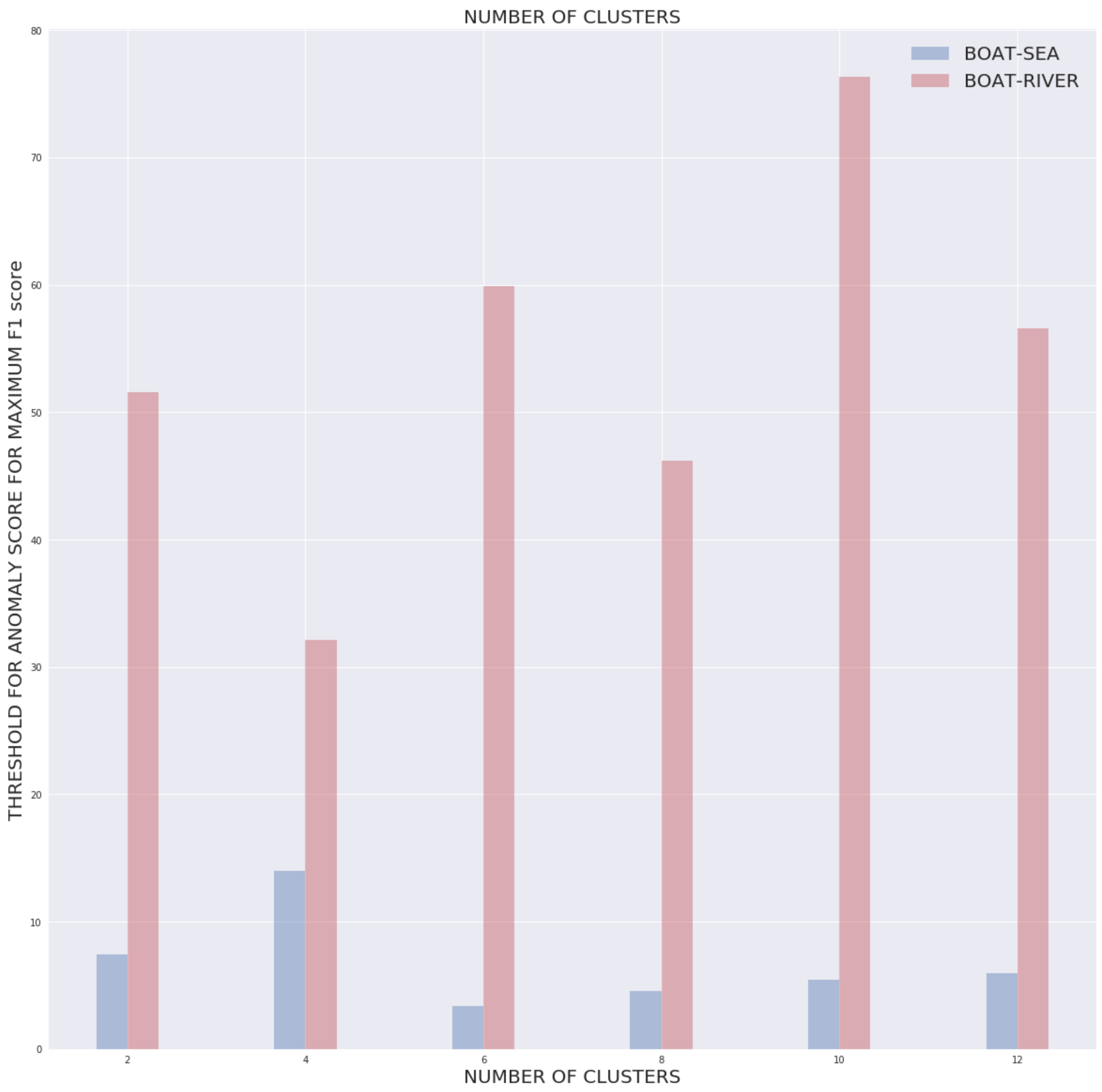


Figure 9–4 – Comparison of the Mahalanobis distance scores for maximum F1 scores in the Boat-River and Boat-Sea dataset for different values of K.

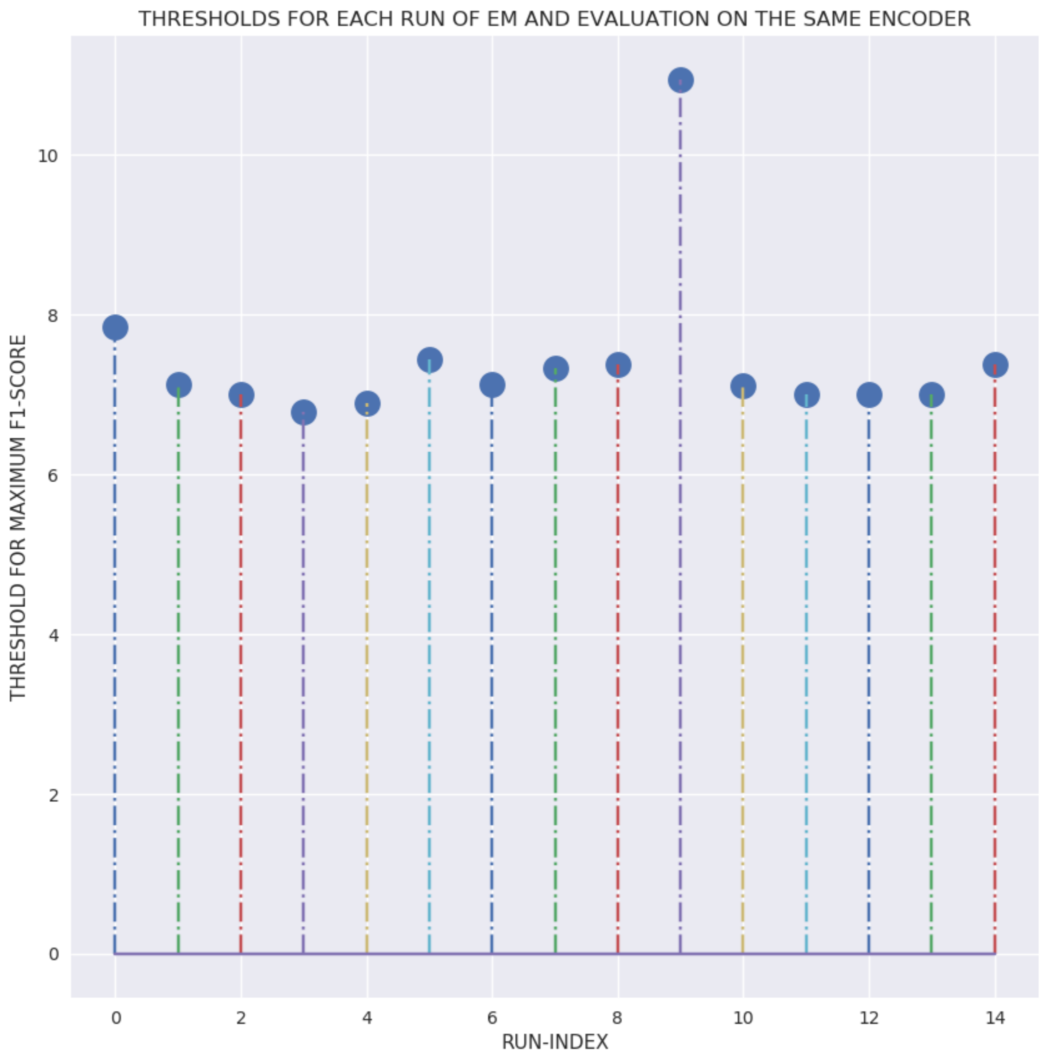


Figure 9–5 – Comparison of the Mahalanobis distance scores for maximum F1 scores in the Boat-River dataset for different runs of the expectation maximization algorithm for fitting the Gaussian mixture model.

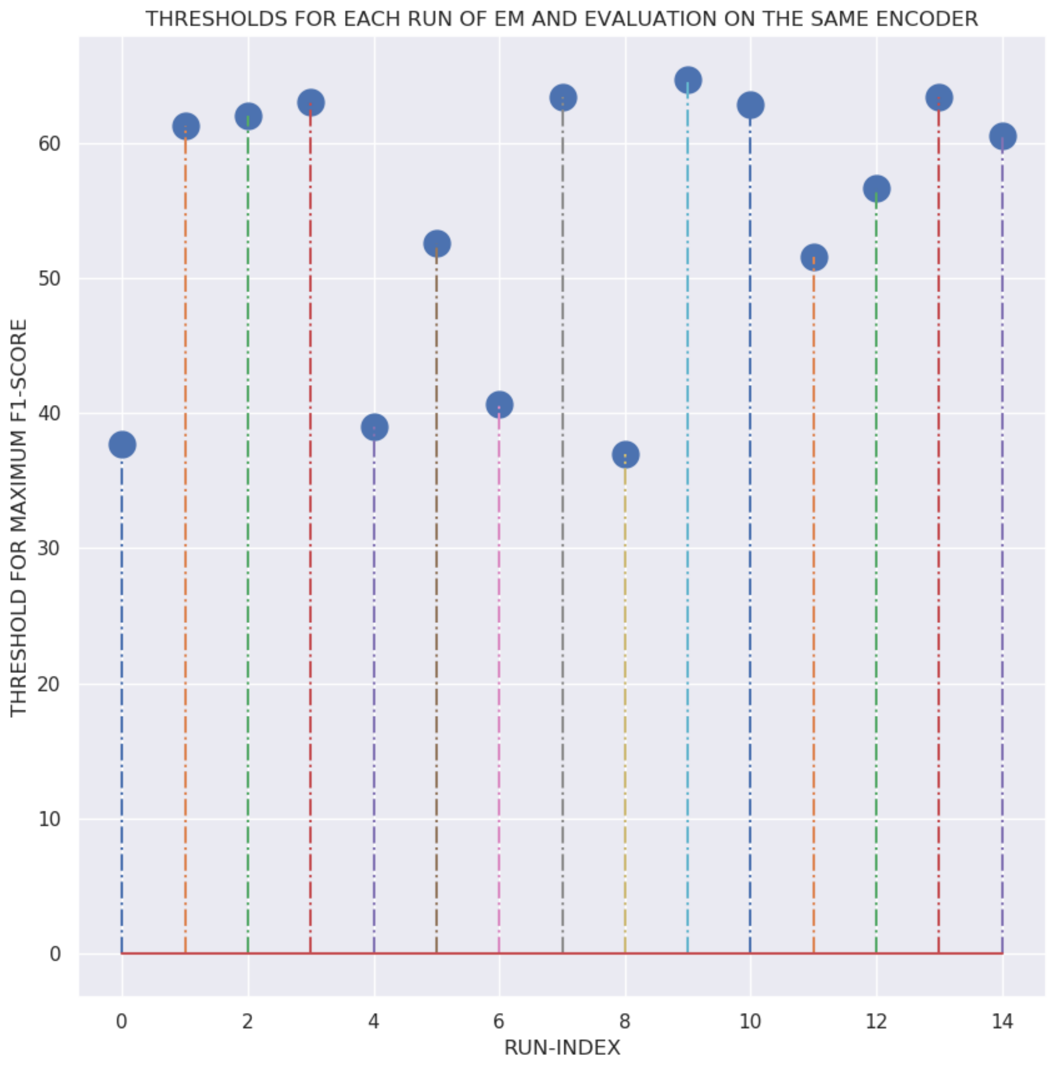


Figure 9–6 – Comparison of the Mahalanobis distance scores for maximum F1 scores in the Boat-Sea dataset for different runs of the expectation maximization algorithm for fitting the Gaussian mixture model.

9.4 Future direction

Video analysis, especially anomaly detection still has a long way to go before reliable, real-time algorithms can be deployed on real world applications. Any solution that depends on a set of hyperparameters has a risk of failure, especially when the data are high dimensional. The hyperparameters heavily affect the performance of the model and the high dimensionality of the data makes debugging a cumbersome task. This problem could be solved by the introduction of efficient hyperparameter search methods.

The field of anomaly detection also carries upon its shoulder, the uncertainty of sampling the videos. To recall the concept: in a dynamic environment, we are quite unsure as to when a particular event starts or ends. In a highly dynamic scene, there can be multiple events and constant motion that can overlap spatially and temporally. For this research, we set the sampling to 5 seconds of video in each spatio-temporal cuboid. Testing the effect of sampling length on a model is quite difficult, considering the time required to train and evaluate the model. We wish to further investigate other directions for solving the problem by avoiding this sampling conundrum. A possible area of exploration might be a fusion model, that looks at the spatial and temporal aspects of the scene separately at first and combining the anomaly scores later. Specifically, the temporal model must sample the scene only when motion exists. If the subject is standing still, the spatial model will be called upon, and when the subject is moving, the temporal model will take over. To avoid extremely long samples, the frame can be divided into grids and each grid can be monitored independently.

It is imperative to have a large natural dataset with good ground-truth annotations for evaluating deep learning models. This work has ameliorated the situation somewhat by providing a framework that can deliver unlimited artificial data with pixel level annotations. However, it is different to testing models on real-world data. The research on generative models has been picking up a lot of momentum recently. We must also investigate whether generative models can be utilized to generate unlimited "fake, but real" data to train deep networks. As this research has concluded, the ground-truth annotations are usually subjective to the observer as well. A clear and concise definition on what is actually considered to be an anomaly must be reached to make the notion fully objective.

REFERENCES

[1] Kdnuggets: Intuitive explanation convolutional neural networks, 2018.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[3] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. In *Readings in Computer Vision*, pages 522–533. Elsevier, 1987.

[4] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18, 2015.

[5] Salman Aslam. Youtube by the numbers, 2018.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[7] Yannick Benezeth, Pierre-Marc Jodoin, Venkatesh Saligrama, and Christophe Rosenberger. Abnormal events detection based on spatio-temporal co-occurences. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.

[8] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[9] Antoni Chan and Nuno Vasconcelos. Ucsd pedestrian database. 2008.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[11] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[12] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357, 2017.

[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] 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.

[15] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*, pages 428–441. Springer, 2006.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[17] Asimenia Dimokranitou. *Adversarial autoencoders for anomalous event detection in images*. PhD thesis, 2017.

[18] Duarte Duque, Henrique Santos, and Paulo Cortez. Prediction of abnormal behaviors for intelligent video surveillance systems. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 362–367. IEEE, 2007.

[19] Yaxiang Fan, Gongjian Wen, Deren Li, Shaohua Qiu, and Martin D Levine. Video anomaly detection and localization via gaussian mixture fully convolutional variational autoencoder. *arXiv preprint arXiv:1805.11223*, 2018.

[20] Homa Foroughi, Aabed Naseri, Alireza Saberi, and Hadi Sadoghi Yazdi. An eigenspace-based approach for human fall detection using integrated time motion image and neural network. In *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pages 1499–1503. IEEE, 2008.

[21] Homa Foroughi, Hamid Reza Pourreza, et al. Intelligent video surveillance for monitoring fall detection of elderly in home environments. In *11th Int. Conf. on Computer and Information Technology*, 2008.

[22] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

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

[24] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[25] 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.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Rolf Herken. The universal turing machine. a half-century survey. 1992.

[28] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[30] Fan Jiang, Ying Wu, and Aggelos K Katsaggelos. A dynamic hierarchical clustering method for trajectory-based unusual video event detection. *IEEE Transactions on Image Processing*, 18(4):907–913, 2009.

[31] B-H Juang and Lawrence R Rabiner. A probabilistic distance measure for hidden markov models. *AT&T technical journal*, 64(2):391–408, 1985.

[32] 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.

[33] Louis Kratz and Ko Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. 2009.

[34] 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.

[35] Gal Lavee, Latifur Khan, and Bhavani Thuraisingham. A framework for a video analysis tool for suspicious event detection. *Multimedia Tools and Applications*, 35(1):109–123, 2007.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] He-Ping Li, Zhan-Yi Hu, Yi-Hong Wu, and Fu-Chao Wu. Behavior modeling and abnormality detection based on semi-supervised learning method. *Ruan Jian Xue Bao(Journal of Software)*, 18(3):527–537, 2007.

[38] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[39] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[40] 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.

[41] 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.

[42] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[44] Alessandro Mecocci, Massimo Pannozzo, and Antonio Fumarola. Automatic detection of anomalous behavioural events for advanced real-time video surveillance. In *Computational Intelligence for Measurement Systems and Applications, 2003. CIMSA'03. 2003 IEEE International Symposium on*, pages 187–192. IEEE, 2003.

[45] Jefferson Ryan Medel and Andreas Savakis. Anomaly detection in video using predictive convolutional long short-term memory networks. *arXiv preprint arXiv:1612.00390*, 2016.

[46] 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.

[47] Marvin Minsky and Seymour Papert. Perceptrons. 1969. *Cited on*, page 1, 1990.

[48] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[49] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[50] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[51] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE, 1998.

[52] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[53] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015.

[54] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[55] 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.

[56] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[57] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[58] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[59] Mohammad Sabokrou, Mahmood Fathy, Mojtaba Hoseini, and Reinhard Klette. Real-time anomaly detection and localization in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 56–62, 2015.

[60] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, and Reinhard Klette. Deep-cascade: cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Transactions on Image Processing*, 26(4):1992–2004, 2017.

[61] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, Zahra Moayed, and Reinhard Klette. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 2018.

[62] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[63] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

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

[65] Jasper Snoek, Jesse Hoey, Liam Stewart, Richard S Zemel, and Alex Mihailidis. Automated detection of unusual events on stairs. *Image and Vision Computing*, 27(1-2):153–166, 2009.

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[67] Statistica. Size of the global video surveillance market from 2009 to 2019, by region (in billion u.s. dollars), 2018.

[68] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[69] Jagannadan Varadarajan and Jean-Marc Odobez. Topic models for scene analysis and abnormality detection. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1338–1345. IEEE, 2009.

[70] 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.

[71] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[72] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[73] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[74] Arnold Wiliem, Vamsi Madasu, Wageeh Boles, and Prasad Yarlagadda. Detecting uncommon trajectories. In *Digital Image Computing: Techniques and Applications*, pages 398–404. IEEE, 2008.

[75] Xinyu Wu, Yongsheng Ou, Huihuan Qian, and Yangsheng Xu. A detection system for human abnormal behavior. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1204–1208. IEEE, 2005.

[76] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[77] 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.

[78] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint arXiv:1610.04794*, 2016.

[79] Y. Yuan, Y. Feng, and X. Lu. Statistical hypothesis detector for abnormal event detection in crowded scenes. *IEEE Transactions on Cybernetics*, 47(11):3597–3608, Nov 2017.

[80] Andrei Zaharescu and Richard Wildes. Anomalous behaviour detection using spatiotemporal oriented energies, subset inclusion histogram comparison and event-driven processing. In *European Conference on Computer Vision*, pages 563–576. Springer, 2010.

[81] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. 2010.

[82] Dong Zhang, Daniel Gatica-Perez, Samy Bengio, and Iain McCowan. Semi-supervised adapted hmms for unusual event detection. In *Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on*, volume 1, pages 611–618. IEEE, 2005.

[83] 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.

[84] 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.

[85] Hua Zhong, Jianbo Shi, and Mirkó Visontai. Detecting unusual activity in video. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.