Motion Event Recognition Using User Feedback

Hannes Jönsson

Examensarbete
2018:E79

Master’s Theses in Mathematical Sciences 2018:E79
ISSN 1404-6342

LUTFMA-3372-2018

Matematik
Matematikcentrum
Lunds universitet
Box 118, 221 00 Lund

http://www.maths.lth.se/

Motion Event Recognition Using User Feedback

Hannes Jönsson
`stv10hjo@student.lu.se`
Advisors: Anders Heyden, Jakob Grundström, Mikael Pendse, Umut Tezduyar Lindskog
Examiner: Kalle Åström

December 20, 2018

Abstract

With high associated costs, both in terms of time and economics, false alarms are a ubiquitous problem in camera surveillance. There exists many methods of lowering the rate of false alarms from motion detection, most of which are based around using static rules and filters such as area of interest, cross-line detection and filtering of swaying objects. This thesis explores the possibilities of using feedback from a user in the form of *examples* to filter new video sequences with detected motion, enabling the suppression of motion alarms that are of no interest to a user. Two approaches to achieve this are presented, a frame-by-frame method employing a one-class support vector machine and a sequence based recurrent relational network which aims to learn to recognize similar and dissimilar video sequences. Both approaches aim to fulfill the requirement of both being able to work with only positive data, and a limited amount of it at that. The report discusses the difficulties and challenges with working with data from only a single class, and the limitations that a limited amount of training data pus on such systems. Both approaches employ transfer learning in that a pre-trained network is used to extract high level abstract features from video frames. The results of experiments run show a surprising level of accuracy with even basic methods under the right circumstances, while also revealing problems with unexpected behavior when encountering some new, unseen data. The report further discusses the problems of establishing how predictable such a system can be, and how communication with a user can be done so as to properly convey how the system can be expected to act. The report makes the conclusions that technically there are indeed possibilities to recognize events in video sequences and enabling the filtering thereof upon example videos, but the problem of understanding what exactly it is in a sequence that a user may want to ignore and to communicate what exactly it is a system has learned to recognize is a very difficult one. The report concludes with suggesting a couple of avenues of future work, both when it comes to the general problem of using feedback from a user, and when it comes to the more technical aspects.

Keywords: One-Class Classification, Relation Network, OSCVM, Motion Detection, Siamese Network, Distance Metric

Acknowledgements

I would like to thank my advisors for their input and support during the thesis work. I am especially thankful for the large amount of help and support I got from Jakob Grundström and Mikael Pendse. They continually took the time to discuss various topics and problems surrounding the thesis, and continually provided me with feedback, recommended interesting articles surrounding my topic and more. I would also like to thank Axis communications for providing the opportunity and the resources required for this thesis.

Contents

1 Introduction

1.1 Background

The presence of motion detection capabilities in cameras today is ubiquitous. By alerting a user, such as a surveillance operator, or triggering an event, such as the recording of video or activating some external system, motion detection can greatly reduce the resources needed to monitor an area. As the use of motion detection strives to increase the effectiveness of a surveillance system by only using resources when something of interest occurs, the accuracy of the motion detection and the way detected motion is handled is crucial to the overall performance of the surveillance system. The occurrences of false positives, detected motion when there is none or detection of motion that isn't of interest, can inhibit the ability to provide the benefits a motion detection system is capable of. False positives can lead to an increased use of the various resources which a surveillance system requires, including storage space and user attention. Especially the latter may lead to *alarm fatigue*, where a user after being exposed to a large number of false alarms becomes less attentive to the alarms and may fail to act in an appropriate manner when a true positive is encountered.

The reduction of false positives in the motion detection and the handling thereof is of great import when an effective surveillance system is to be achieved. There are several approaches to solving this problem existing today, with many of them already implemented in products on the market. In Axis cameras, which have been used in this thesis, several types of software aiming to solve this problem are available. This includes the filtering of small and short lived objects, the detection of swaying objects such as flags or foliage, *cross line* detection triggering when a moving object crosses a certain line, areas of inclusion/exclusion and more. The main characteristic of all these approaches is that they let a user select certain static parameters, such as tolerances and thresholds for when a motion alarm should be triggered or designating a certain area as off-limits and triggering an alarm when motion is detected in such an area, and filtering detected motion based upon these parameters.

This thesis aims to explore a different approach to solving the problem of reducing unwanted motion detection alarms, based upon the idea of letting a user present an example of a motion event which might trigger a motion detection alarm, and having the system automatically adjust to the needs of the user.

1.2 Problem formulation

The main goal of this thesis is the investigation of approaches to minimizing false positives from the motion detection system based on user feedback. False positives in the context of this thesis are defined as any motion event a user is uninterested in. It is thus not limited to the detection of non-existent motion, such as that which a shadow or a reflection moving through the surveillance scene might give rise to, but also actual motion events that for some reason are uninteresting to the user. User feedback is defined as consisting of an example video, or image sequence, representing the unwanted motion event.

The problem to be solved can be formulated as: Is it possible to devise a system such that it can recognize any event based upon an example, or possibly a few examples, provided by a user as feedback, and thus make it possible to avoid triggering a motion alarm should a new motion event be considered being of the same type as that which the user deems uninteresting? In answering this question, several other questions have to be addressed, for example; How can the amount of examples a user needs to provide be minimized? How should the relation between several concurrent events be viewed? What does an event really encompass?

1.3 Scope of the thesis

The focus of this thesis project will be answering the questions described in the problem formulation. It will not dwell on the details of the actual motion detection mechanics, nor will it include the general infrastructure for presenting motion events to a user and letting the user select such events as examples. In the interest of time and resource constraints, these matters will be considered outside the scope of the project. In keeping with this definition of the scope as well as to improve the reproducibility of any results produced during the project, pre-recorded video will be used as a basis for experiments conducted, rather than real time streamed video. A detailed description of the datasets used for experiments is provided in the relevant sections of the report.

What is meant by the term *motion event* should also be defined. Without further clarification, such a term is ambiguous at best. For example, if a user wants to exclude the occurrence of a garbage truck passing through an area that is under surveillance, what exactly does this mean? If the garbage truck first passes one way and then returns in the other direction, are these two distinct events? Should the event be considered as consisting of a garbage truck passing through the field of view, or is a garbage truck driving from left to right in the scene a separate event compared to a garbage truck driving from right to left in the scene? It is easy to transfer a discussion of such terminology into a more philosophical domain, and thus some hard definitions need to be set up. In the interest of avoiding making the basis of this thesis dependent on such definitions, what constitutes a motion event is defined as per the experiment setup. A detailed explanation of what exactly is meant by a motion event and the correct recognition thereof in the context of an experiment is provided in the description of each experiment setup. Further discussion of this topic and its implications can be found in the conclusions section of this report.

The data used during the experiment mainly consists of publicly available datasets. Purpose-made datasets constructed for this thesis are also used in order to provide data for performance on test setups more closely resembling real world use-cases, compared to pre-made datasets.

The scope of this thesis includes the presentation of two different approaches to solving the problem at hand. For each of these approaches, an explanation of the theory behind it, a motivation in the selection of the approach and a discussion of the implementation details is provided. These discussions will not include low-level implementation details, but rather focus on the broader, architectural choices.

The scope thus includes the answering of questions such as:

- What are the challenges in basing a motion event filtering system around feedback from a user?
- Is it possible to recognize *events* in sequences based upon example sequences of such events?
- Can a network pre-trained on static images be used in a transfer learning scheme for use on video sequences?
- Can a distance metric learning network be implemented in a recurrent manner for use on sequences?
- How well does a basic support vector machine approach deal with sequences of data?
- How does a support vector machine based system compare with a distance learning one when it comes to distinguishing between sequences?

2 Related work

There are several works relating to this area, though not with the same approach as that presented here. Some of these works are presented in this section.

2.1 Sensor fusion

In a previous master's thesis conducted at Axis, Bilski investigated the use of sensor fusion in order to reduce the frequency of false motion detections[1]. The approach presented in that thesis focuses on the addition of data gathered from a radar unit in order to validate motion detection alarms. The idea here is to use data from a radar in combination with motion data from a video stream in order to filter out detected motions caused by artifacts other than actual moving objects.

The results presented in the thesis show that the use of radar data for validation can lower the frequency of false positives, as a radar is unable to detect artifacts such as shadows and reflections and needs an actual object in the scene which can reflect the electromagnetic waves. The capabilities of a radar unit is also invariant with respect to the lighting conditions of the surveilled scene, enabling the detection performance to be consistent irregardless of time of day and other aspects affecting the lighting. Indeed, the results suggest that with only detection of actual moving objects as a measure of performance, a radar can in itself perform as good as, if not better than, motion detection conducted solely on a video stream. Of course, using only the radar data not much else can be said about the detected motion other than the approximate size of the reflecting side of the object, its distance and velocity. Any specific visual characteristics need to be gathered from the video stream [1].

The focus of this approach is the minimization of false alarms by filtering for real, physical objects. As such, it seems an effective method, but it does not lend itself to adapting to a specific users requirements, which might include the filtering of specific physical objects which actually does move.

2.2 Object classification

Another master's thesis conducted recently at Axis investigates the use of an image classifier to help with the reduction of false positives in the motion detection [2]. The idea is to select only relevant classes, which the thesis group into a category described as *human activities*, and only report detected motion if it should stem from such a *human activities.* Motion from objects classified as not belonging to one of the classes in this group, such as foliage swaying in the wind, a cast shadow moving through the scene or a reflection off of a windscreen of a car are thus ignored.

The thesis presents a method of doing this which leverages a pre-made, unspecified program, which given JPEG images as input outputs predicted labels for the content of the image. Input images were cropped in order the have them contain only the moving objects which should be analyzed and avoid the inclusion of background objects that could affect the classification of the object in the image. Each image was then assigned a class membership, and this class was then checked against the set of classes of objects defined as being *human activities*, which consisted of the following classes: Person, Bicycle, motorbike, Car and Bus.

The performance of the approach is measured by how good a job the classifier does at classifying such cropped images representing the moving objects. The report states that $\sim$85% of all objects were classified correctly. This figure held true between both objects defined as part of the *human activities* set and as part of the *non-human activities* set. 92.6 % of *human activities* were correctly classified as such and would then give raise to a correct alarm. The remaining 7.4 % where incorrectly filtered as uninteresting, non-human activities. Among the moving objects that should have been ignored as non-human activities, 93.2 % were correctly identified as such and thus filtered, while 6.8 % resulted in false positives. The report also states that while 7.4 % of moving objects with a true class belonging to *human activities* were mistakenly filtered, this does not mean that the true false positive rate in the alarms would be 7.4 %, since this number is partly a result of the way in which the moving objects are tracked and the images containing them are cropped.

The method presented in this thesis shows a lot of promise with regard to lowering the frequency of false positives. It does also lend itself somewhat to customization to a users requirements, in way of letting a user choose which object classes are to be ignored. This approach covers many scenarios, but it does not let the user select an arbitrary event, which might not be able to be represented as a single class in the classifier, for exclusion. For example, an event such as a vehicle traveling in one direction could be seen as a distinct event when compared to the same vehicle traveling in the opposite direction, but both scenarios would result in the same object classification, making it impossible to distinguish between them just from the classification.

3 Theory

3.1 The One-Class problem

The general classification problem is a problem that has been studied thoroughly throughout the history of machine learning. Most who have encountered machine learning in some form have probably heard about the Iris flower dataset, originally presented by Ronald Fisher in 1936[3], which through many a blogpost and YouTube video serves as something of a *hello world* of machine learning. The dataset records five attributes each from 150 flowers, belonging to three different species with 50 examples each. This is a very typical multi-class classification setup, where the objective is to learn a method of classifying a flower, based upon its attributes, as one of the given set of species. In contrast to this multi-class classification problem, one might instead ask not 'to which species (class) does this flower belong?', but instead 'is this a flower or not?'. This is a vastly different problem. In the case of having different classes, each consisting of a different flower, defined by a set number of measurable attributes, it is feasible that a dataset of reasonable size can cover most of the possible cases. In the case of defining a class such as 'not a flower', this encompasses every conceivable entity that is not a flower, and creating a dataset that would cover such a vast space of possibilities is impossible. A more reasonable approach would be to view the data for the class 'not a flower' as absent, and instead see what can be done with the data that is there, for example descriptions of flowers. With data from only one of the classes ('flower') and none from the other ('not a flower'), the problem is now a one-class classification problem.

With a one-class classification problem, the goal has switched from the creation of boundaries between different classes based upon examples belonging to each class, to deciding if an example is part of a the target class or not. This can be achieved in a couple of different ways, two of which will be discussed here.

The first approach is to create a decision boundary around a single class, aiming to contain as many examples of this class as possible while keeping the probability of including examples outside of this class as low as possible. This is not as easy as constructing a decision boundary for the multi-class classification problem, and the requirements on the dataset and the methods employed can be expected to be higher[4]. An example of this approach is one-class support vector machines[5], which will be discussed further in section 3.2.

The second approach is to make comparisons between different examples. This would, in the case of the flower-or-not example problem mean comparing any given data with a known flower, and assigning a value describing the similarity between the target class, in this case the flower, and the data for which class belonging is inferred. This would mean creating a system which can learn to distinguish between similar and dissimilar pairs of data. Examples of this approach can be found in the form of *siamese networks*[6] and relational networks[7]. This approach will be discussed more thoroughly in section 3.5.

3.2 One-Class Support Vector Machines

In their most basic form, Support Vector Machines (SVMs) can be described as machine learning tools that can be used for solving classification or regression problems. A basic setup would be that given a dataset $\{(x_i, y_i), .., (x_n, y_n)\}$ where x_i is a data point and y_i is a label corresponding to this data point which describes its class, a SVM would strive to create a boundary in the space which the data points occupy such that data points sharing the same label are grouped together. This would then let the SVM *predict* which label belongs to a new, previously unseen data point. The SVM is also able to separate otherwise inseparable data points by project them into a higher dimensionality space using a non-linear function, enabling it to create a boundary, a hyperplane, separating data points which it could otherwise not. This hyperplane is adjusted so that the *margin*, the shortest distance between a data point on each side of the hyperplane and the plane itself, is maximized.

This basic approach can be adapted to the one-class classification problem, in which data may or may not be associated with a known label, but there is only access to training data belonging to the known label at training time, making it impossible to create a boundary *between* examples of different classes. Instead the similarity between data points known to belong to the known label is described, for example in the form of a spherical area around the data points, to which the distance of new data points can be calculated and thus their label inferred. Tax and Duin describes this as given a training dataset $\{x_i, .., x_n\}$, a *hypersphere* with a center $\mathbf{a}$ and a radius R is created around it[5]. The construction of this hypersphere involves a minimization problem in which the volume of the sphere is reduced my decreasing R^2 so it is as small as possible while still containing all of the training set. This can be described by the error function to minimize:

$$F(R, \mathbf{a}) = R^2 \tag{1}$$

with the constraints that:

$$||x_i - \mathbf{a}||^2 \leq R^2 \text{ for i = 1, .., n} \tag{2}$$

Allowance can be given to outliers, so that some data points can be excluded from the sphere in order to avoid making it too large. This can be described by a margin ξ_i and an associated penalty variable C, which controls the trade-off between the volume of the hypersphere and the errors encountered when excluding positive samples as the sphere is reduced in volume. With the inclusion of this margin for outliers, the error function can be described as:

$$F(R, \mathbf{a}) = R^2 + C\sum_{i=1}^{n} \xi_i \tag{3}$$

with the constraints that:

$$||x_i - \mathbf{a}||^2 \leq R^2 + \xi_i \text{ for i = 1, .., n} \tag{4}$$

that is to say, the squared absolute distance between a given data point x_i and the center of the sphere $\mathbf{a}$ must be less than the squared radius of the sphere plus the tolerance margin ξ_i. A new data point can then be classified in regard to its position, with it either being within the hypersphere, thus deemed as positive and part of the target class, or outside it. There are also alternatives to a strict spherical volume, such as polynomial kernels or radial basis function kernels, which serve to enable the creation of more flexible separation functions[5]. The radial basis function kernel (RBF) is what is used in this thesis and is defined as:

$$K(x, x') = exp\left(- \frac{||x - x'||^2}{2\sigma^2} \right) \tag{5}$$

The choice of this kernel for the particular problems faced in this thesis is based on studying the performance using different available kernels, with RFB resulting in the best performance.

The use of One-class support vector machines is a straight forward approach which can give great results with a low computational performance cost. The main challenge is to find a format for the data such that a decision boundary actually can be formed around it, separating it from data points not belonging to the class to be identified. As the complexity of the data grows, the ability to group examples of data deemed to belong to the same class together tightly as well as the ability to separate examples of data belonging to different classes diminishes. This is known as *the curse of dimensionality.* A 3-channel color image with the dimensions 224 by 224 pixels (a typical input size for InceptionV3[8]) can be described with 150528 different values. It can thus be represented with a 150528 dimension vector, a large number indeed. While a clustering approach may work well on a smaller toy example, such as MNIST images in gray-scale with dimensions of 28 by 28 pixels thus having only 784 dimensions, it quickly breaks down when the number of dimensions reaches higher numbers. In the results section, we will see the ramifications of this in detail.

3.3 Artificial Neural Networks

Many of the techniques used in this thesis are based on the capabilities of artificial neural networks. They are both used in order to extract high level abstract features from images in a video stream, as well as a basis for the relation net module which learns to compare different feature representations. This section will present a brief overview of the workings of artificial neural networks in order to facilitate easier explanations of the respective system.

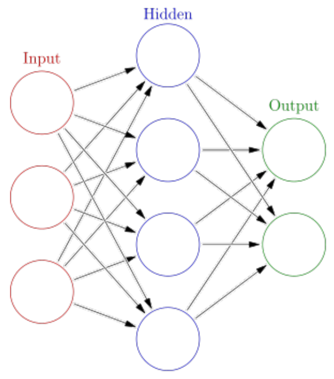


Figure 1: Illustration of a basic ANN.

The term *artificial neural network*, ANN, comes from the fact that such networks are loosely inspired by what we know about the workings of organic brains such as our own. Our brains consist of large, complex interconnected networks of neurons, which is, at least on the surface, similar to the artificial neural network which consists of several interconnected computational nodes. At the most basic level, an ANN strives to approximate a function mapping an input to an output in a certain manner [9]. This mapping of input to output can cover a large variety of tasks, including classifying a given input as belonging to one of a set of classes, giving a control signal to a system (such as learning to play computer games [10]), or even transforming an image, for example reducing noise in a noisy image. This mapping of input to output is conducted by sending signals from node to node in the network, with the signal being controlled by the characteristics of the input and by the parameters of the nodes it passes through. An ANN learns the input to output mapping by tuning these parameters associated with its nodes in a process called training. Training is generally conducted with the use of data similar to the actual problem to solve. In most cases, this training data consists of examples of input as well as the desired output, and the network will then adjust its parameters until the inferred output of the network matches the desired output. An explanation of the actual workings of the tuning of the parameters of the network during this training is outside the scope of this report, but a curious reader will find a thorough explanation in Goodfellow et al.'s Deep Learning book[9], or in Grant Sanderson's excellent video series on his YouTube channel 3Blue1Brown[11].

Artificial neural networks come in several different flavors. The most basic form are known as feed-forward networks, in which data flows from the input nodes to the output nodes, being operated on in accordance to the parameters of the network. The nodes are often organized into layers, with layers in between the input and output layers being known as hidden layers. In addition to the feed-forward type of network, two other more specialized forms of artificial neural networks are employed in this thesis, namely convolutional neural networks and recurrent neural networks. Convolutional neural networks use a mathematic operation known as a convolution as it operates on the data that flows through it, and is suited for data that takes on a grid like form, such as images and time-series data[9]. In a practical sense, the use of convolutional neural networks can be easily motivated with a simple example. If we consider an image as the input for an artificial neural network, the complexity at the input layer increases rapidly with the size of the image. An image of size 224 by 224 pixels with 3 color channels (a standard input for networks such as InceptionV3) fed into a layer of 1024 nodes will result in more than 150 million parameters to tune during training, and that is just for one layer. With the use of convolutions, the size of the data can be reduced while retaining information, as the convolutional network learns to extract *abstractions* from the input. A convolutional network can achieve this with just a few parameters, often ranging in the thousands for a single layer.

Recurrent neural networks are networks specialized for operating on sequences. This includes sequences of words, such as might be the case in classifying text or image sequences such as might be the case when operating on video. The use of the term *recurrent* comes from the fact that the result of an operation on one part of a sequence affects the operation conducted on the following parts of the sequence. In such a way, the way in which a recurrent neural network handles a word in a sentence is affected not only by the networks parameters and the word itself, but also which word or words came before it in the sentence. Thus a recurrent neural network can take the temporal aspect of data into account [9]. In the methods described in this thesis report, variations and combinations of these types of artificial neural networks are employed.

3.4 Transfer Learning

Convolutional neural networks are well suited to operate on data consisting of images, which is exactly what this thesis requires, but they do however require large amounts data to train, and the training is time consuming. To get around these constraints, a scheme called transfer learning can be employed. The basic idea is to leverage a network which has been rigorously trained on a large, diverse dataset which hopefully encompasses the domains which are of interest for this thesis, so that the training of a complex network does not have to be done from scratch. Using such a method has been shown to produce astoundingly good results on tasks such as image classification, even in areas differing from the ones on which the network has originally been trained[12].

In the scope of this thesis, transfer learning is used by employing InceptionV3[8] and MobileNet[13] with weights pre-trained on the ImageNet dataset[14]. These networks are trained for the task of image classification with a thousand classes for a long duration of time. For the purposes of transfer learning, the output of an intermediate, hidden, layer is used with the hope that this output represents learned high level abstractions of the input images. These high level abstractions will enable the images to be represented with a lower dimensionality when compared to the raw pixel values, while still retaining useful information about the contents of the image.

3.5 Relation Net

As discussed in section 3.1, one a approach to solving the one-class classification problem is to compare data and assign a value describing its similarity or dissimilarity to known reference data. Thus previously unseen data can be recognized as either, if it's similar to an example of the target class, part of the target class, or not. This can be seen as a problem of learning a distance metric with which to compare different data points. One method for conducting such recognition is the use of a Relation Net. A Relation Net represents an approach to few-shot learning, where a classifier must learn to recognize new classes with the use of only a few examples of each class. A Relation Net, as described by Sung et al. in [7] works by using two modules, an embedding module f_ϕ and an relation module g_θ. The embedding module's purpose is to encode input data into feature mappings, while the relation module's purpose is to produce a similarity score between 0 and 1 for any pair of features representing two data objects. This can be expressed as

$$r_{i,j} = g_\theta(f_\phi(o_i), f_\phi(o_j)) \tag{6}$$

where $r_{i,j}$ is the similarity score between the two objects o_i and o_j.

As training data a set of N classes, with each class being represented by K_N examples, containing data with a corresponding label map of which data points are similar to which is used. From this set of N classes and N by N similarity scores, C classes are selected to make out a *sample set*, while N - C classes are selected as a *query set*. To set the problem up as a one-shot problem rather than a few shot problem, the feature mappings can be pooled if a class is represented by more than one example. These feature sets $S = \{(x_i, y_i)\}, i = \{1..K \cdot C\}$ and $Q = (x_j, y_j), j = \{1..K \cdot (N - C)\}$ called the sample set and query set respectively, simulates the overall setup where the system at test time uses a support set and a test set occupying the same label space. The training is conducted by selecting pairs from the two sets and using the labels to determine if they are similar or not. Similarity is labeled as 1 and dissimilarity is labeled as 0, and the system is trained with mean squared error as a loss function. This is motivated by the problem not actually being a binary classification problem with the two classes similar and dissimilar, but instead being a regression problem with possible values in the range $\{0, 1\}$.

The approach presented in [7] uses a end-to-end training regime, which allows the encoding module to learn representations which are suited for use by the relation module, rather than, for example, learning representations suited for reconstructing the input as is the case in an autoencoder. It is not difficult to argue that learning to encode a given input in such a way that the encoding best suits the problem at hand is preferable, but such an approach does come with a few drawbacks. The main drawback with training an encoder with a loss functions centered on the final relation score is the requirement of labeled data. For any input pair $\{x_i, y_i\}$, a ground truth relation score needs to be assigned in order for a loss value to be calculated in regard to the inferred relation score. This can severely limit the available selection of training data. Even with a dataset suited for such a task, the fact that we need the dataset to be somewhat balanced puts further constraints on the approach. For example, if we produce a toy dataset containing the numbers in the range [0..9] and select pairs $\{x_i, y_i\}$ from this set such that they receive a relation score $r(\{x_i, y_i\}) = 1$ if $x_i == y_i$ and $r(\{x_i, y_i\}) = 0$ if $x_i \neq y_i$, only 9 out of the 81 possible pairs would be labeled with a relation score of 1, while the remaining 72 pairs would be labeled with a relation score of 0. This would severely imbalance the dataset, something which would inhibit the training of the model. In order to balance the dataset, only 18 out of the possible pairs could be selected. This will apply to any dataset containing more than two types of examples. A solution for this is to instead train the encoding module separately on as large a dataset as possible, and not caring about the relation between the training examples. This makes it possible to train on an (with regard to relational scores) unlabeled dataset, increasing the amount of data suitable for training. By using a pre-trained network geared towards image-analyzing in a transfer learning scheme (see section 3.4) as a primary embedding module, this effect can also be achieved. The same trade-off is true in this case, namely that while the embeddings themselves are optimized for another problem than the one we want to solve, they are still trained on a vastly larger dataset, thus their increased ability to describe an input in a meaningful way should enable the extracted features to be used to great effect none the less, as shown in [12].

This approach is very much similar to a *siamese network*[6]. A main difference with a *siamese network* is that it instead of concatenating two separate inputs and inserting them into a net as one input, it handles two inputs separately in two parallel but identical networks. Having two *paths* with identical weights enables the network to not over-fit in such a way that the output of the network is dependent on which of two inputs goes through which path, rather than the make up of the input data itself. In [6] such a network is used to learn a distance metric, which in principle is the same thing as learning a similarity metric.

3.6 Metrics

An extremely important factor when evaluating different approaches to solve any problem is the metrics that are employed. Both *what* is measured and *how* it is measured is of critical import, if reasonable conclusions are to be drawn from conducted experiments. The different approaches presented in this thesis are not always directly comparable using a simple measure of performance that can be applied to all experiment setups. Because of this, this section will present the different metrics used during training and testing of the models used to evaluate the approaches.

When measuring the performance of a classifier which can make a binary choice, as is the case for a one class classifier such as a OCSVM which classifies examples presented to it as either a member of the positive target class or not, it is possible to measure the performance in several ways. Performance can be measured by way of accuracy as in the total number of correct classifications relative to the total number of examples tested on:

$$\text{Accuracy } = \frac{\text{Number of correct classifications}}{\text{Total number of classifications}} \tag{7}$$

This metric suffers when used to evaluate a classifier classifying a dataset with a severe class imbalance, as is the case with the datasets used in this thesis. If, as is the case here, a large portion of the dataset belongs to the negative class, that is, is not of the target class, the classifier can still achieve high accuracy by solving the classification problem trivially and simply classifying all test examples as negative, non-members of the target class. The same problem holds true for the precision metric, which calculates the rate of false positives:

$$\text{Precision} = \frac{\text{True positives among the examples classified as positive}}{\text{All examples classified as positive}} \tag{8}$$

By being extremely discriminatory, a classifier can achieve a very low false positive rate, which will result in a high precision even if very few true positive examples are correctly identified.

Another metric which suffers with an imbalanced dataset is recall. Recall measures the rate at which a model can detect positives with respect to the number of existing true positives. If recall is to be maximized the opposite problem can occur, compared to accuracy or precision; the model might choose to classify all examples as positive, thus classifying all the true positives as such.

$$\text{Recall} = \frac{\text{Examples classified as positive}}{\text{All true positives}} \tag{9}$$

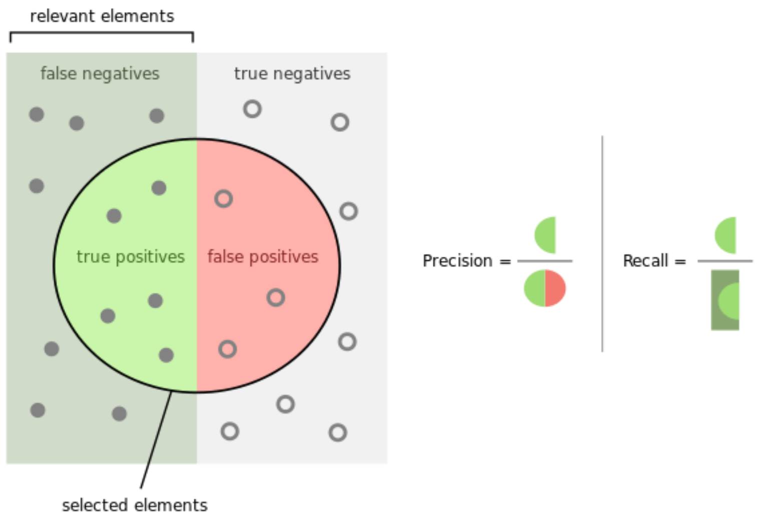


Figure 2: Illustration of the relation between true & false negatives and precision and recall.

To balance these metrics a metric called F_1 score can be employed. F_1 score is defined as:

$$\mathrm{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{10}$$

By including both recall and precision in a single metric, a balance between detecting true positives as such while still avoiding false positives can be achieved. This makes this metric suitable for measuring the performance of a classifier such as OCSVM.

The selection of which metric to use as a loss function when training a neural network is very important. There exists a large array of pre-defined loss functions in the Keras[15] and TensorFlow[16] frameworks, suitable for different types of problems. In the case of a network producing a score for a given input, a loss function which uses a measure of how far from the desired score the predicted score is would be suitable. One such loss function uses the *mean squared error*, which is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{11}$$

that is, the average difference between the predicted score and the true score for a given test set. In the case of inferring relational scores for a set of sequence pairs, this loss function would result in the average squared difference between the ground truth relation score (1 or 0 for similar and dissimilar respectively) and the inferred relation score. The calculation of accuracy for the relational network is based upon rounding the inferred relation score to the closest integer (0 or 1), and comparing with the ground truth value. The difference can then be averaged over the test set to gauge the accuracy. This can be expressed more formally as:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}(1 - |y_i - [\hat{y}_i]|) \tag{12}$$

where $[\hat{y}_i]$ is the predicted relation score, rounded to the closest integer. This is in contrast to the OCSVM based approaches, which infer class membership on a frame-by-frame basis after which a decision has to be made for the video as a whole.

4 Implementation Details

4.1 Frameworks

Various machine learning frameworks and tools have been used in this thesis. Most code was written in the Python programming language[17]. For image and video processing, scikit-image[18] and scikit-video[19] have been used. For clustering approaches such as OCSVM, the scikit-learn framework was used[20]. For setting up a feature extraction network based on InceptionV3 and mobile net with weights pre-trained on ImageNet, Keras[15] was used as a front end with TensorFlow[16] as back end. Keras was also used to construct the relational network based models.

Scikit-learn is a machine learning library for the python programming language, which provides easy to use features such as different clustering methods, tools for solving various regression problems etc. For the purposes of this thesis, it is mainly the various clustering methods that were employed, primarily one-class support vector machines, OCSVMs.

Keras is a wrapper that works with various different backends such as Theano and TensorFlow. For the purposes of this thesis, TensorFlow was used as a back end. The purpose of using a wrapper such as Keras was to speed up and streamline the development of different neural network structures. The selection of TensorFlow as back end was due to its large developer community, which makes it easier to find valuable tips, tricks and help with solving various problems that arise during development.

4.2 Datasets

During testing, training, validation and testing of the two approaches described in this thesis, a few different datasets where used. These datasets are described in this section.

4.2.1 MNIST

During the initial proof-of-concept phase, the MNIST dataset of handwritten digits[21] were used, both in its original and in a modified form. The MNIST dataset consists of 60,000 training images and 10,000 testing images, depicting handwritten digits in the range of 0 to 9. A dataset based on a modified form of the MNIST dataset, called *moving MNIST* was also used. This dataset was generated as per [22], and consists of sequences of images, with each sequence depicting two digits moving about in a random manner. The parameters of the generation of the sequences were selected so that 120 images, each being a gray-scale 64 by 64 pixel image containing two digits, were produced.

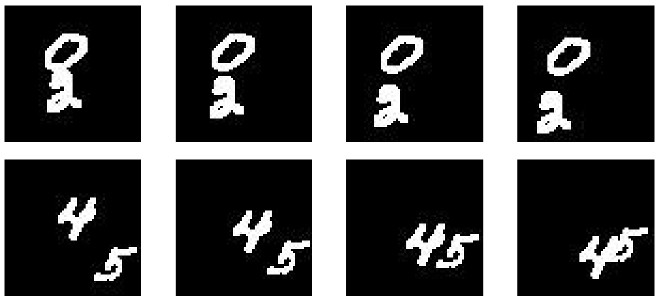


Figure 3: Example frames from two sequences, one containing a 0 and a 2, the other containing a 4 and a 5.

4.2.2 UCF101

The main dataset used in this thesis was the UCF101 action recognition dataset. This is a dataset consisting of 13320 video clips depicting 101 different types of human actions, gathered from YouTube [23]. The clips are MPEG-4 encoded with a resolution of 320 by 240 pixels. The per frame image quality is quite low, but it has been shown that image compression has a limited effect on the performance of an image recognition system, at least in the context of face recognition[24]. The 101 actions are divided into 25 groups each, with each group being represented by 4 to 7 video clips. The clips representing each group all share something in common, for example the action portrayed in a group might be conducted by the same actor or actors. The main purpose of the dataset is to enable training and validation of action recognition systems, with recognition of an action being defined as classifying the category to which a video clip belongs correctly. For the purposes of this thesis, the UCF101 dataset is used in a slightly different manner. The division of the action categories into subgroups, each being represented by clips sharing attributes, is the main reason for the use of this dataset, as it enables a labeling scheme where video clips belonging to the same action category and group can be labeled as being similar, while clips belonging to different groups or action categories can be labeled as dissimilar. It is thus possible to instead of aiming to classifying a video clip as belonging to a certain action category, aim to infer whether a video clip belongs to a certain such subgroup. This is a way of translating motion event recognition into a domain where the performance of such a motion event recognition system can be trained and tested on a large, pre-existing dataset. The use of such a large, public dataset should also allow for reproducible results.

4.2.3 Other data

Further data was gathered with the use of Axis cameras set up in a bike storage room. This data was not extensive enough to be used as training data or to make larger validation data sets, but can instead serve as more realistic examples of real world use cases. As the the different approaches can not be directly compared by a metric such as the validation accuracy during training time, or by a loss function value (which is not present in the case of OCSVM), this dataset also serves the purpose of enabling a more direct comparison of performance between approaches. The videos recorded in this dataset consists of various actions in the bike storage room, such as the entry and exit of one or several actors, actors carrying objects like boxes or bags, actors entering or exiting with bikes etc. The exact setup for what constitutes a similar and dissimilar event in this dataset is discussed in the discussion of specific test setups.

4.3 Representations - Feature extraction

In order to make use of the transfer learning scheme described in section 3.4, high-level features are abstracted from the images that make up the video sequences. Features are extracted on a frame by frame basis with the use of two different neural networks with pre-trained weights. The two networks are MobileNet[13] and InceptionV3[8], both of which are used without the fully connected top layers, and with an added layer of max pooling. This results in a feature vector per image frame, with a length of 1024 for MobileNet and 2048 for InceptionV3. A larger feature vector can hold more information, but is not necessarily better, as it increases the dimensionality of the data, which can lead in an increase in the requirements of training data in order to achieve good performance[24]. These per-frame representations are used both on a frame-by-frame basis by the OCSVM approach, and fed together as a sequence in the case of the relation net based on extracted features. As the lower number of features in the MobileNet produced features did not seem to adversely affect the performance of the models tested, these features were the ones mainly used due to their lower dimensionality and decreased use of space.

4.4 One-Class Support Vector Machines

The One-Class Support Vector Machines, OCSVM, model is based upon the idea of working with features extracted on a frame-by-frame basis from the video sequences. The point of using a One-Class SVM approach is to solve the problem of only having access to positive examples to train on, and thus the training of the OCVSM model is done on hand-picked video sequences representing positive examples of a motion event. In order to keep the clusters tight each type of motion event, here defined in the manner described in section 4.2, is assigned to its own model. While this introduces an upper limit to the amount of different classes of motion events that can be represented by a system, as the number of models grows, the models are quite small and not very computationally expensive to run, making it possible to facilitate the running of a decent number of simultaneous models and thus types of motion events. Examining how the performance of an OCSVM based system scales with the number of included classes has been left outside the scope of this thesis, and the focus has instead been on examining the behavior of the models when exposed to different type of motion events.

The actual implementation was done with the use of scikit-learn[20], which includes an OCSVM module with several options enabling a user to tweak it for the problem at hand. The options available include the type of kernel used, where the choice in this thesis fell on the rbf-kernel, and the two parameters *gamma* and *nu*. The *nu* parameter is an upper bound on the tolerated amount of training errors, and has the practical effect of 'shrinking' the decision boundary as it increases in magnitude. The *gamma* parameter is the kernel coefficient, which also adjusts the decision boundary such that it becomes tighter as the parameter increases in magnitude.

Several models were trained, based upon several different classes of video sequences. Different numbers of training examples for the single class were tested. A rudimentary grid-search was implemented in order to conduct parameter tuning of the model. The tuning was conducted with F_1-score as the metric for which to optimize. The parameters tuned were 'nu' and 'gamma'. Training was conducted in such a way that for a given positive class, a model was constructed with the use of the given examples of the class and two values for the parameters to be tuned. Evaluation was then done on a randomly sampled set of frames from a test set of video sequences containing both videos of the positive class and videos of other, negative classes. The recall-, precision- and F_1-score were then calculated based upon the inferred classification made on these test sets of frames. The process was repeated for each possible parameter value within the given range, and only the models which gave the best F_1-score on the test set were saved.

The output of the model given an input frame is a classification value signifying whether the model deems the frame to be part of the set of frames it has seen, representing the class the model is to recognize, or not. As such, performance can be measured in two ways; either by counting the number of correctly and incorrectly classified frames, or by defining a different selection method where a sequence of frames as a whole is classified as either part of the class trained on or not. In the case of making a decision on the sequence as a whole, this can be achieved with the use of a threshold, set up in such a way that if a certain number of consecutive frames in a sequence presented to the model is deemed not to be of the class trained on, the sequence as a whole is classified as negative. This threshold can itself be seen as a parameter that can be tuned in order to achieve a certain balance between the risk of including false positives and ability to detect true positives. A threshold of 40 consecutive frames was tested to give an idea of how such a method would perform.

4.5 Relational Network

The relational network model is based on the idea discussed in section 3.5, but the implementation used here differs in several key factors when compared to what is described in [7] as well as compared to a siamese network such as that described in [6]. First and foremost, the objects between which the relation score is to be calculated in this thesis are video sequences, not still images. This has two major implications; Each sequence consists of several images, making it difficult to create a straight forward embedding module consisting of convolutional layers as described in the paper. Each sequence also has a temporal aspect, that is, how the content of the images in the sequence change over the sequence is an important factor in what the sequence actually portrays. To take these two implications into consideration, the relation net based idea is implemented using recurrent neural networks [9].

Using a relational or siamese approach to constructing a network which at the same time takes the temporal aspect into account requires some thorough design work. In the end, the sequential nature of the input should be transformed into a single output, being the measure of similarity between the two input sequences. This is what is known as a *many-to-one* network, and requires a recurrent layer to handle all the elements in a sequence and produce a single output. This leads to the need to decide *where* this transformation from sequence to a single element takes place. It could for example be handled by each of the two identical parts in the relational network, and merging the output from these, no longer being of a sequential nature. It could also be done at a stage further into the network. Placing the *many-to-one* part of the network after the output of the two identical nets are merged means that the merging of the output from the two *arms* of the network will be done at each time step in a sequence, meaning that the result of such a merge will take effect at the time step level. Placing the *many-to-one* part before the merging of the two arms will instead mean that the merging will affect a single element, into which the temporal aspects of a sequence has been embedded. During the implementation of such a system, both approaches were tested, and putting the recurrent *many-to-one* part before the two arms are merged seemed to perform better. This can possibly be explained intuitively as the identical recurrent networks being responsible for learning an embedding, such that sequences deemed similar are embedded in a similar manner while dissimilar sequences are embedded in a dissimilar manner, while the latter part of the network is responsible for interpreting the resulting embeddings and producing a similarity score for them. The merging can be conducted in several different ways. In [7], it consist of a simple concatenation of the embedded features, resulting in a feature vector of twice the length of the features for the individual sequences. In [6], the element wise difference of the two embeddings is calculated and given as input to the latter part of the network. In the implementation presented here, the merging is conducted as follows:

$$\{1 - |x_i - y_i|\} \text{ where } x_i \in \text{ left tensor and } y_i \in \text{ right tensor} \tag{13}$$

This should mean that the network is forced to learn embeddings such that two similar sequences produce a merged tensor of values close to 1, while dissimilar sequences produce a merged tensor of values, all of which are not close to 1. As the loss function, MSE was used. To evaluate the network, both a test split of the data used for training (moving MNIST and UCF101) is used, as well as other data gathered for this thesis as described in the datasets section of the report is used. The training was monitored with the help of TensorBoard[25].

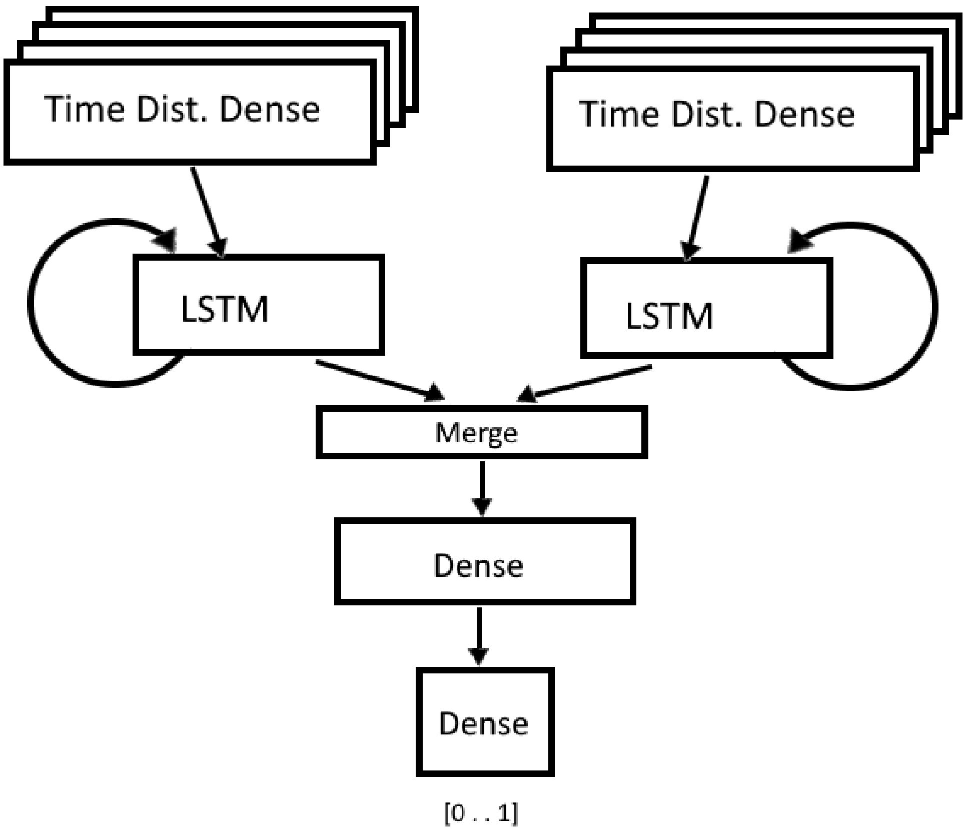


Figure 4: Schematic of the relational network architecture.

5 Results and Evaluation

Experiments were conducted with a few goals in mind. Initial experiments were conducted in order to evaluate the feasibility of the chosen approaches. These initial experiments were conducted on a smaller scale and on datasets not necessarily representing the whole domain of the problem. After the feasibility of the approaches was established, further experiments where run on larger datasets in order to evaluate the methods. This section of the report describes the specifics of each of the experiments in regard to what data was used, and how the metrics were defined, as well as presents the results from the experiments. The section is subdivided with subsections assigned to each of the approaches examined, discussing the particularities of the experiment setups for each of them.

5.1 Experiment setup

5.1.1 Model validation

Validation of the different models was primarily conducted with the use of validation data from the same domain as the training data. The reason for this is the availability of reasonable amounts of validation data in the public datasets used. Performance on validation data taken from the same dataset as the training data can be a good indicator of the performance of the system, but it does not guarantee the performance level in a more realistic use case scenario. Because of this, further evaluation was conducted with the use of datasets gathered for this express purpose.

5.1.2 OCSVM

During training, the OCSVM model is tuned with regard to the F_1 score, as described in section 3.6. In the case of experiments being run on non-video based data, such as MNIST images, the score is measured per *in-class* or *target class*, that is, per digit being considered as the class to be identified. Precision, recall and accuracy are also measured. In the case of experiments run on data from video sequences, the score is measured on a frame-by-frame basis, where a positive is defined as a frame belonging to a video sequences in the set of positive test examples and a negative being a frame belonging to any other video sequence. Frames are selected at random during the validation phase of the parameter tuning. When measuring the number of consecutive frames classified as in or out of the target class, the frames are handled in a sequential, ordered manner. In the discussion, the ramifications of the frame-by-frame approach and ways to deal with them is discussed.

5.1.3 Relation network

The setup used for the relational network is very different to the setup used for the OCSVM. Since the model does not strive to create a decision boundary in order to be able to distinguish between a positive target class and a negative class, but instead to be able to correctly score similar video sequences with a high relation score and dissimilar video sequences with a low relation score, the training is no longer conducted with the use of examples of a single class, used to construct a model per class. Instead, sequences are paired and based on their scene given a ground truth relation score based on the scene to which they belong. In order to avoid creating a trivially easy training and validation set, as well as avoid creating an impossible one, the correct hardness of the similar and dissimilar pairs has to be achieved. This is very much similar to the reasoning used when selecting good triplets for example in the case of face recognition[24]. The definition of *too easy* and *too hard* is done in a straight forward way: A dissimilar pair is considered too easy if the constituent sequences belong not only to different scenes, but also to different action categories. Similar sequences, which are defined as being of the same category and the same scene, are not allowed to actually be the same sequence either. Instead, only sequences portraying the same scene of the same category but not being the same example is allowed to be paired as similar, while only sequences from the same category but not the same scene are allowed to be paired as dissimilar. Example image frames from such sequences can be seen in figure 5.

The relational network is trained and tested in to phases; first on moving MNIST sequences generated for this purpose, and thereafter on video sequences from the UCF101 dataset. The purpose of this is to establish the methods viability on a more controlled dataset before evaluating it on real life video

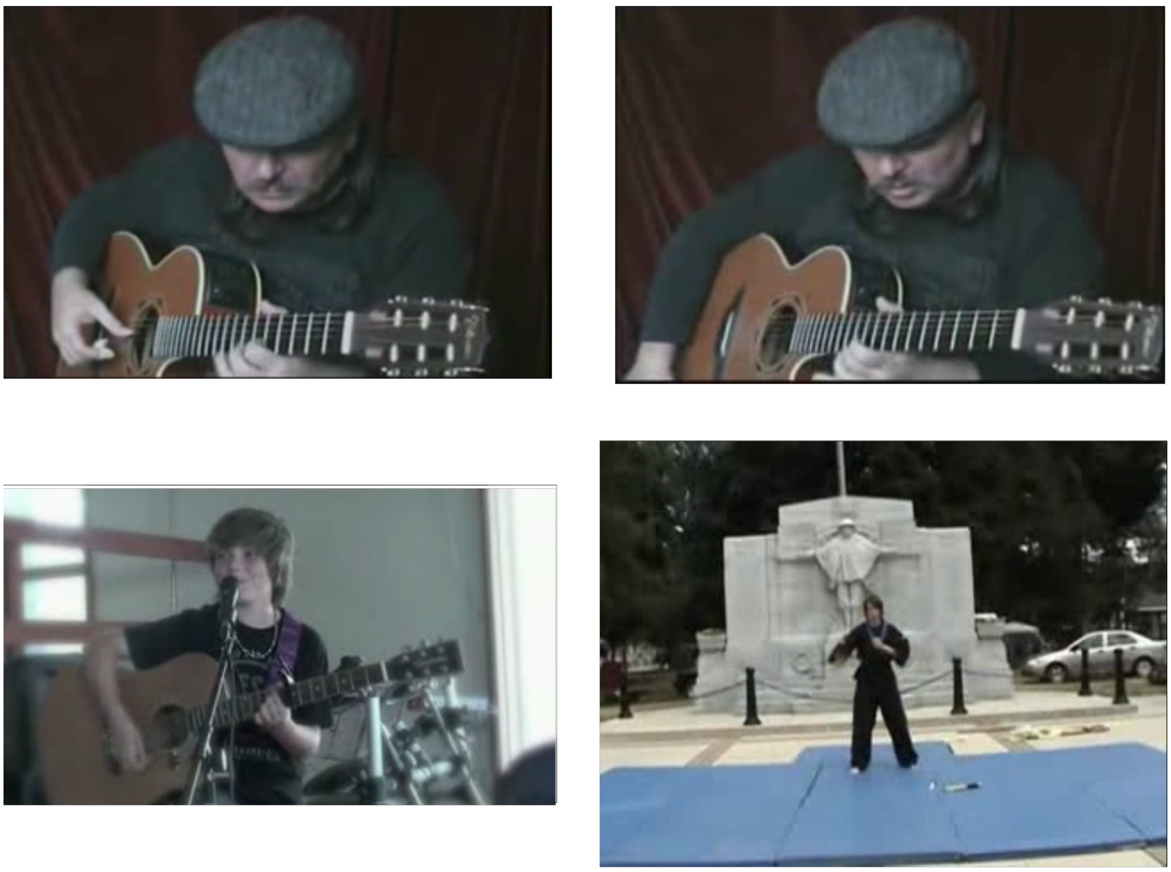


Figure 5: Frames from four different videos. In respect to the top left image, the top right is considered similar, bottom left a good choice of a dissimilar video and bottom right a too easy dissimilar video.

data. The training of the network is done with respect to the validation loss, as defined by equation 10 in section 3.7.

5.1.4 Use case evaluation

To evaluate the performance of the systems in regard to a more realistic use case scenario, the dataset gathered from the bike room at the Axis offices was used. For each possible setup, evaluation was done on the performance of the systems. The setups differentiated in that for the relation net approaches, the performance was measured as how well two given sequence could be recognized as either being of the same type or not, while for the OCSVM approach the measurement was simply on how well a model constructed for a single such motion event could differentiate between other instances of the same event and other events. In the case of the OCSVM approach, several such models were produced in order to cover a larger amount of possible cases.

5.2 Experiment results

5.2.1 OCSVM

5.2.1.1 MNIST feasibility study

In order to validate a proof of concept for the OCSVM approach, a test was run on the MNIST dataset with a one-class configuration. In this test, a 100 training images of a single type of digit was presented to the system, which then learned to distinguish between images depicting this digit and images depicting another number. Parameter tuning was conducted for the *gamma* and *nu* parameters.

The validation of the proof-of-concept setup of a one-class SVM system on the MNIST dataset gave promising results, given the test setup. The performance varied widely depending on which digit was designated as the target class, as can be seen for example in the case of training on images of the digits 2, 7, 4 and 9. In all of these cases, the models displayed a tendency to mis-classify certain digits as being of the target class. In the cases of models trained on 7s 4s and 9s, the problem was mutual. In all cases, the models had a tendency to think that any of these numbers were of the target class. This is not that surprising as handwritten 7s, 4s and 9s do share a lot in common. A somewhat more surprising problem was encountered by the model trained on images of 2s, which miss-classified the majority of 1s.

As for what can be said for the general performance of these models, it is important to consider that each of them are trained only on images of a single digit, and only on a fraction of the amount of training images at that. This makes it very difficult to compare it with most classifiers run on the MNIST dataset, which generally are multi-class classifiers trained on the whole training dataset, comprised of 60000 images rather then the 100 used here. The state-of-the-art error rate for such a classifier is 0.21%[26], while the average error rate for the 10 models presented here is 13.42%. Another factor to take into consideration when evaluating these numbers is the fact that the models are working with raw pixel data from 32 by 32 pixel images, rather than some form of feature representation extracted with convolutional layers. The reason for this is the low amount of information contained in a grays scale 32 by 32 pixel image, which can be held by a 784 dimensional array of the color intensity values. Extracting features with the help of an InceptionV3 network produces features with a dimensionality of 2048. Such an increase in dimensionality worsens the performance of an OCSVM model, and any other SVM model. Working with such extracted feature representations of the images is worthwhile first after the amount of information contained in an image grows beyond the size of the feature representation, as is the case when working on image frames from the UCF101 dataset, for example.

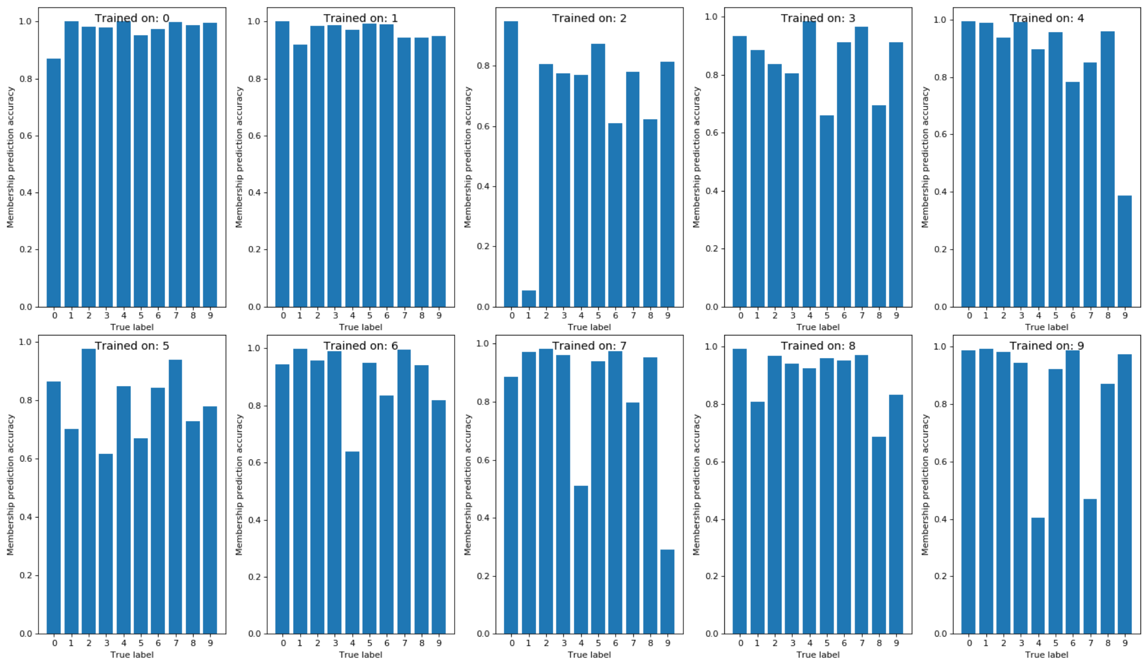


Figure 6: Results of one class classification validation tests run on the MNIST dataset. The accuracy represents the rate at which each model classified images of digits as being of the target class or not correctly.

Trained on	Precision	Recall	F1 score
Digit: 0	0.742	0.875	0.673
Digit: 1	0.847	0.936	0.885
Digit: 2	0.297	0.708	0.412
Digit: 3	0.368	0.768	0.498
Digit: 4	0.495	0.850	0.625
Digit: 5	0.268	0.705	0.389
Digit: 6	0.551	0.796	0.651
Digit: 7	0.286	0.863	0.430
Digit: 8	0.337	0.786	0.472
Digit: 9	0.353	0.877	0.503

Figure 7: Precision, Recall and F1 score for the models trained on the MNIST dataset.

5.2.1.2 Moving MNIST

To evaluate the OCSVM model with regard to sequential data, the *moving MNIST*[22] dataset was generated and used according to the description in section 4.2.1 and 5.1.2. For each pair of digits, only four sequences of 120 frames each were used as training data. The F_1 score as well as the AUROC score can be seen to vary quite a bit depending on which type of sequence was selected as the positive target class. There can be several potential explanations for this; for example, the number of negative example sequences containing one of the digits used in the positive test sequences, the similarity between different digits used as well as the way digits overlap with each other during the sequences. When using sequences with a 1 and a 5 as the target class, the score is quite low and the accuracy of the model is not much better than what could be expected were the frames in the sequences classified at random. When using sequences with pairs of 3 and 8 or 6 and 0 the result is much better. It that these digits simply appear more distinct to the model. The differences in precision, recall and F_1 score can be seen in the table below. Another metric which can be studied for the classification of sequences is the number of consecutive frames classified as being *not* of the positive target class. The average number of consecutive frames being classified as true and false positives can also be seen in the table.

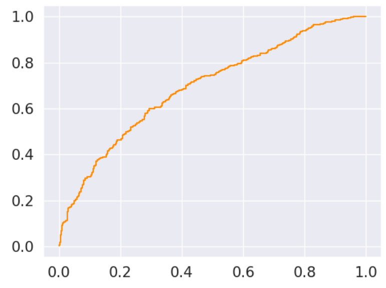
(a) Metrics

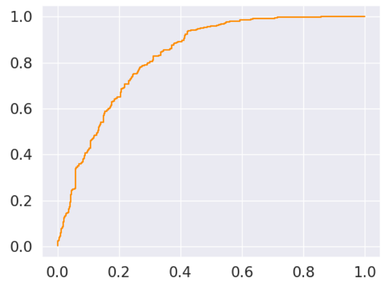
Trained on	Precision	Recall	F_1
1 & 5	0.574	0.810	0.672
3 & 8	0.756	0.723	0.740
6 & 0	0.885	0.867	0.876

(b) Consecutive frames

Trained on:	true pos:	true neg:
1 & 5	4	12
3 & 8	8	34
6 & 3	3	49

Table 1:
(a): Precision, Recall and F1 score for models trained on different types of sequences from the moving MNIST dataset.
(b): Number of consecutive frames classified as negative, averaged over the number of sequences.


(a) Model trained on sequences of 1's and 5's.


(b) Model trained on sequences of 3's and 8's.

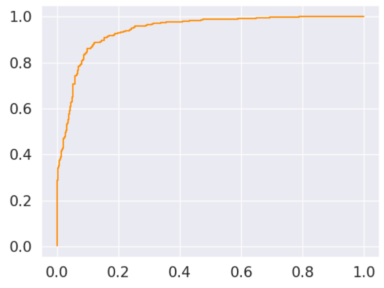

(c) Model trained on sequences of 6's and 0's.

Figure 8: The ROC curve for models trained on examples from three different classes of sequences.

5.2.1.3 UCF101

As for the experiments run on the UCF101 dataset, the OCSVM model was once again trained with parameter tuning of the *gamma* and *nu* parameters, with regard to F_1 score. As target class, a number of training scenes were selected from the UCF dataset on random, all being of the same *scene* as described in section 4.2, and a number of examples from these scenes were then selected as positive training examples. Since the models are trained specifically for each set of positive examples, there is not much that can be said of how generalizable the results are, as they vary widely from test case to test case. To be able to ascertain more exact and interesting information about the different models with their different setups of training data, a selection of models with the corresponding positive training and validation data were made. This selection included both models that performed well and models that performed poorly.

General performance measurements for models trained on different videos:

Model trained on:	Precision	Recall	F1score
Guitar 16	0.96	0.98	0.97
Nunchucks 25	1.00	0.90	0.95
Golf Swing 4	1.00	0.82	0.90
Salsa Spin 1	1.00	0.74	0.85
Pole Vault 2	1.00	0.92	0.96
Band Marching 19	1.00	0.00	0.01
Diving 23	0.25	0.00	0.02
Jumping Jack 15	1.00	0.96	0.98

Figure 9: Precision, recall and F1 metrics for the various models trained on different classes of scenes selected from the UCF101 dataset.

When examining the performance figures for the models, it is important to remember that the numbers can't be compared directly to other systems running tests on the UCF101 dataset. Just as is the case with the MNIST dataset, the UCF101 dataset is mainly used for multi-class classification problems, more specifically for the classification of different human actions. What is of interest here is instead the recognition of similarity between different scenes, enabling the classification of them as either part of the target class or not. This problem is harder partly since the total number of classes available in the dataset is higher, by a factor of 25, which in many cases reduces the inter-class differences, and because the training data available is smaller in size. The difficulty is also higher in accordance with the general reasoning about one class classification problems, in that the absence of training examples from the negative class increases the requirements on the training data[4]. Another consideration that has to be made is that all the metrics are measured on a frame-by-frame basis. While this may be the most consistent way of measuring performance, it is not necessarily the best way to infer the class membership of an entire video sequence. Such a decision might be better to base on the inferred class membership of a group of consecutive frames. By counting the number of consecutive frames inferred to be a member of a certain class, the class of the whole video can be decided and the performance measured in this way. If such a decision is to be based on a certain number of consecutive frames, that number will greatly affect the behavior of the system. For the purposes of this thesis, intensive testing to find an optimal number for such a threshold will be omitted, and as an example the choice has been made to set that limit to 40 frames, which seem to give a decent performance on most tested sequences. Scenes where more than 40 consecutive frames are considered not part of the target class, and thus being of a negative class membership, are classified as not part of the target class. The inferred class of the video as a whole is then compared to the ground truth, which is based upon the scene which the model has been trained on. The performance of a sample of 8 models trained on different scenes can be seen in figure 10.

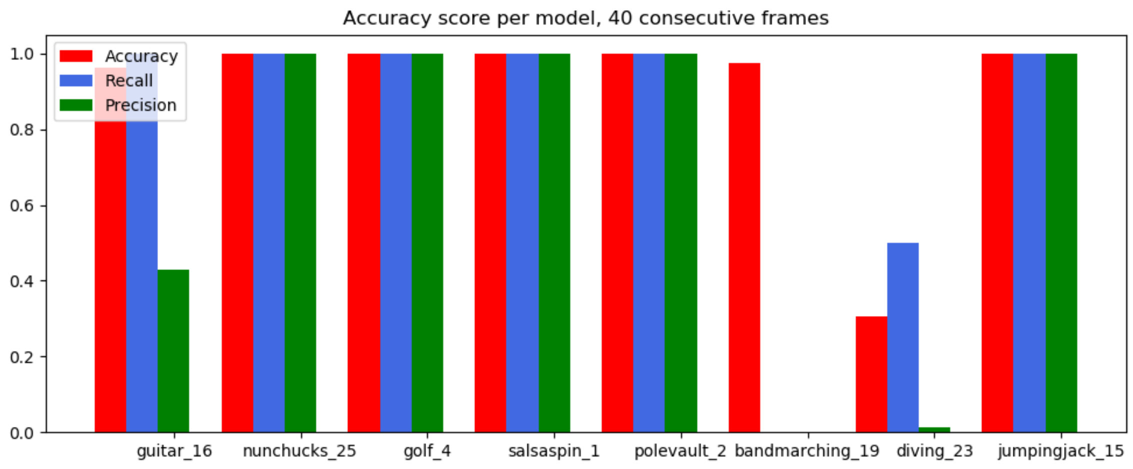


Figure 10: The Accuracy, precision and recall for eight different models, trained on different scenes. The numbers are for videos classified as a whole, using a threshold of 40 consecutive frames before classifying a video as not of the target class.

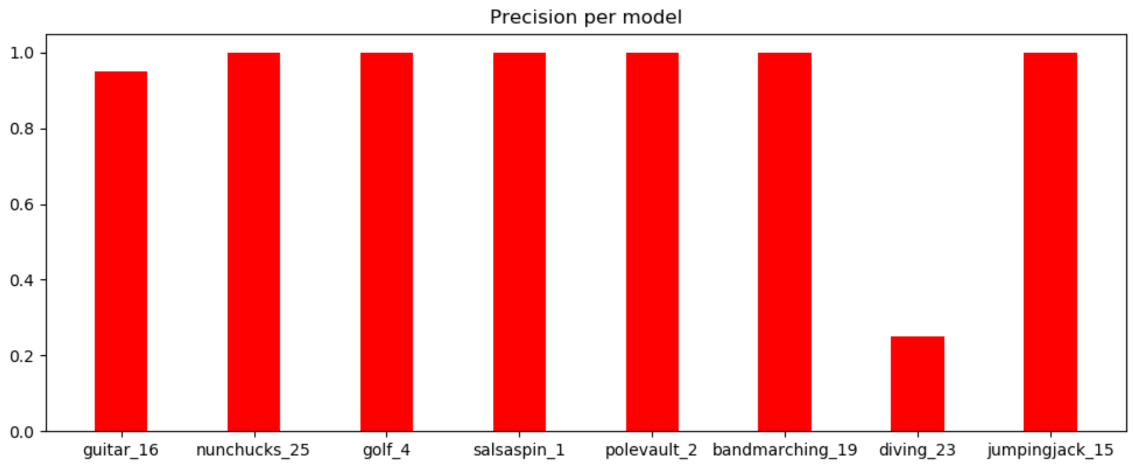


Figure 11: Precision for 8 different models, trained on different scenes.

By plotting the precision, recall and F_1 scores individually for a group of different models trained on different sequences, a better picture of how different models trained on different scenes perform can be achieved. As can be seen in these plots, achieving high precision, that is, a low number of false positives, is trivial when looking at frames individually. This can especially be seen in the case of the model trained on the scene *bandmarching 19*, where the model achieves a high precision score, but it fails almost completely at the recall score. At the frame level, the only thing needed is for the model

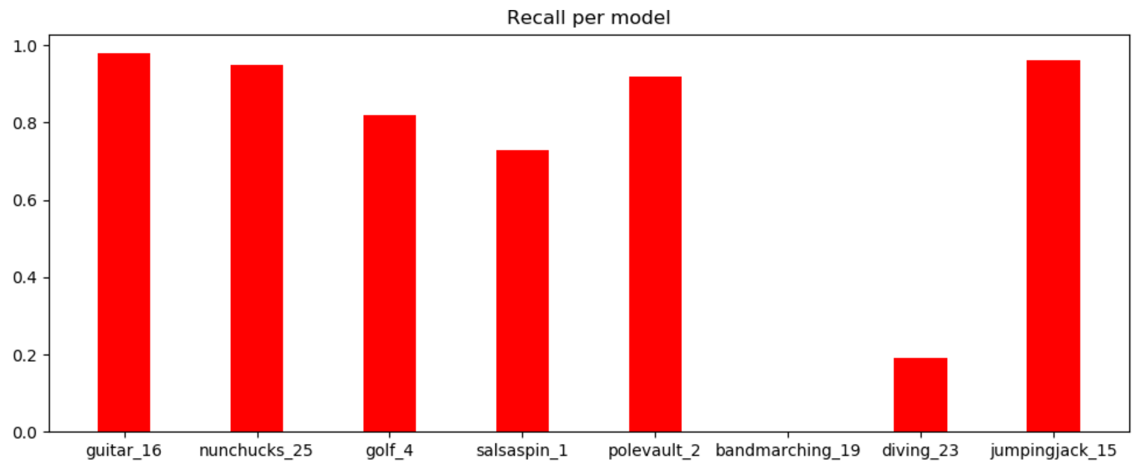


Figure 12: Recall for 8 different models, trained on different scenes.

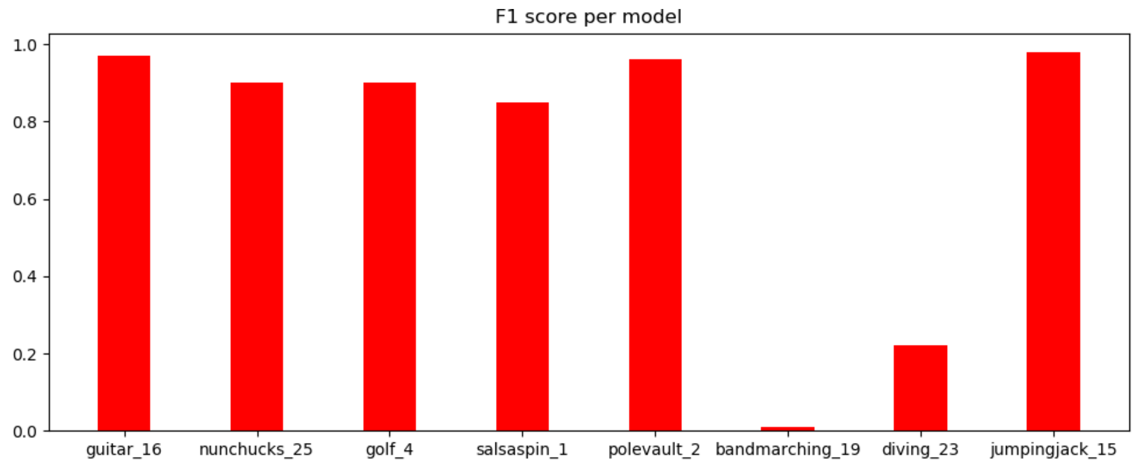


Figure 13: F_1 scores for 8 different models, trained on different scenes.

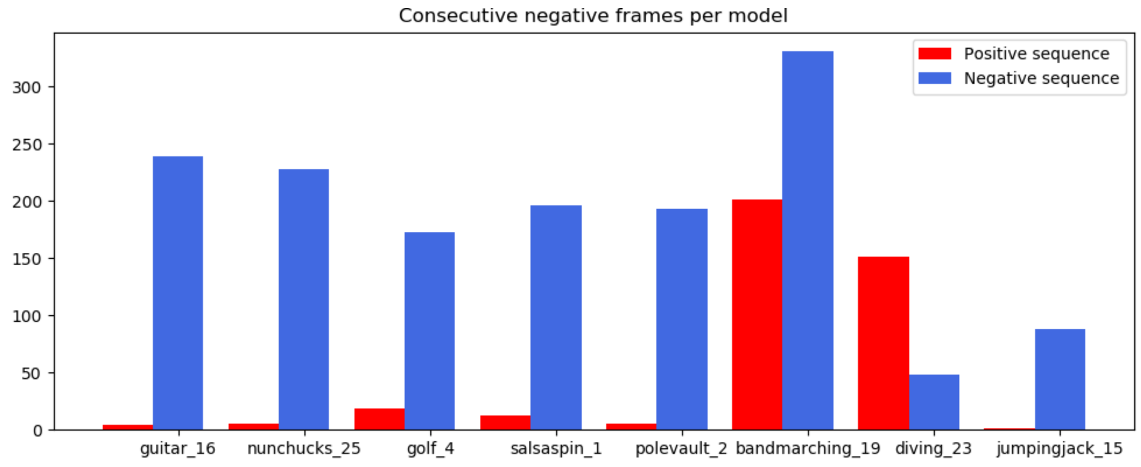


Figure 14: Number of consecutive frames classified as being negative. Averages for truly positive and truly negative examples per model, each trained on a different scene.

to be highly discriminatory as for which frames to classify as being members of the positive target class. When using consecutive frames to classify video sequences as a whole, a low threshold for the number of frames needed to be classified as negative before the video as a whole is classified as such will also achieve a low number of false positives. The problem is that this does not guarantee that the model will be able to correctly classify truly positive examples at a satisfactory rate, or even any at all. This is reflected by the relatively low recall score, as recall is a measure of the rate at which true positives are classified as such. Of course, a dataset of this structure, being constituted of more negative than positive examples, will naturally gravitate toward a tendency to prioritize avoiding false positives as this is easier given the structure of the data. Because of the abundance of negative examples in the test data and the limited amount of positive test examples, the test data easily becomes skewed as well. This is the reasoning for limiting the number of negative test examples used per model when evaluating them, as well the use of F_1 score when training the model, as it represents a more balanced metric in which the ability to correctly classify true positives is also taken into account.

When looking at the number of frames classified as negative in positive respectively negative sequences for the different models, it is possible to see a clear tendency to correctly classify more consecutive frames as negative in the case of a negative sequence, than in the case of a positive sequence. This makes it possible to, at least in the case of some of the scenes, use the model to with a high degree of certainty distinguish between scenes of the same type and scenes of different types. In many cases though, the performance leaves much to be desired. Some models even tend to classify more consecutive frames as negative for the truly positive sequences, when compared to the negative ones. A look at what type of scenes that produce this behavior can give some insight to why this is. A breakdown of number of consecutive frames classified as negative, on average, for the previously used eight models can be seen in figure 14. From this figure, it is clear that is the same two models which have shown performance issues earlier, that still lack in performance.

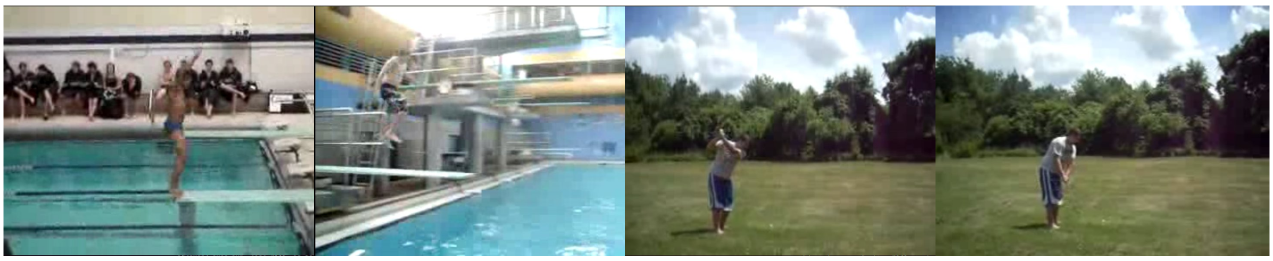


Figure 15: Two leftmost images: Classified as the same scene in the UCF101 dataset, resulting in very poor performance. Two rightmost images: Classified as the same scene in the CUF101 dataset, resulting model has a high performance.

When studying a scene from which a high performing model can be trained and comparing it with a scene on which it seems hard to train a well performing model, some things become apparent. In the case of models that fail in regard to the recall rate, that is, they have a hard time recognizing frames from the same class of scene, it is clear that at the way similar and dissimilar scenes are defined in respect to how the data looks is to blame. For example, if different examples of the *same* scene are filmed from different angles, the UCF101 dataset will sometimes still group them together as one scene. The different angle will however make the scene appear as something completely different to the OCSVM model, and it will thus not be able to recognize two examples of the same scene filmed from two different angles as similar. In some cases, scenes that are dissimilar in many more ways still are also grouped together. An example of this can be seen in figure 15.

5.2.1.4 Use Case Evaluation

With a new model being created for each positive class to be discovered, the ability to function across multiple domains is not a requirement on the model as such. Nevertheless the format of the data itself, in this case the format of the video sequences that make up the UCF101 dataset, can have an effect on the performance of the system. Because of this, a smaller test with the video sequences recorded for the purposes of this thesis was conducted. Sample frames from this dataset can be seen in figure 16, and results from training on a sequence from this dataset can be seen in figure 17

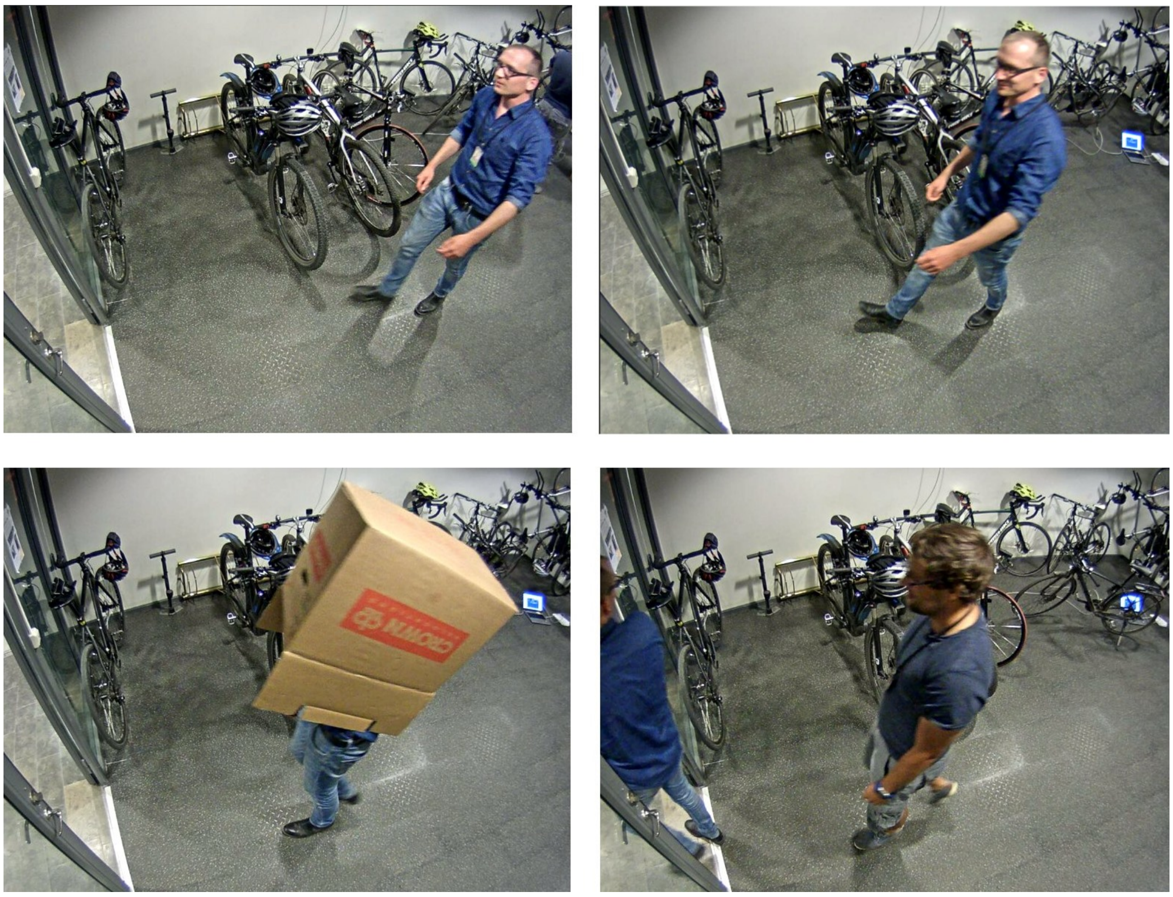
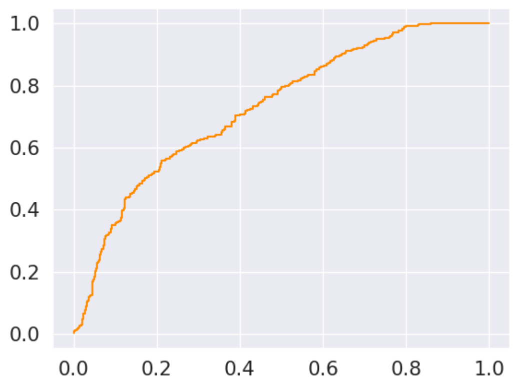


Figure 16: Upper left: Frame from target class video example. Upper right: Frame from positive example. Lower left: Frame from negative example. Lower right: Frame from negative example.



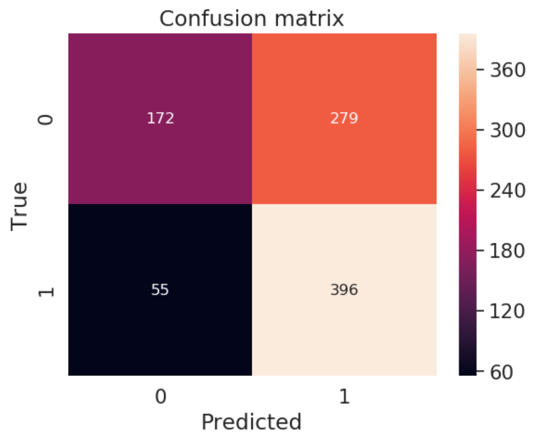


Figure 17: Left: ROC curve and confusion matrix for the model, trained on four example videos. Right: Confusion matrix for the predicted class belonging

As can be seen from the confusion matrix in figure 17, a lot of the frames from examples from the target class were correctly classified as such. However, the amount of false positives on a frame-by-frame basis is very high. This is not very strange, when one considers that most of the scene remains the same between the sequences, and the sequences do have a few frames in the beginning with only the background, making a number of falsely classified frames unavoidable. A look at the number of consecutively classified frames gives a better picture of the performance. Though the dataset is very small, the confusion matrix in figure 18 shows that the number of false negatives shown in figure 17 need not be a problem. Rather, the number of false positives is the main problem with this particular model for this particular threshold of consecutive frames. Changing the threshold of consecutive frames needed for a sequence to be classified as outside of the target class could result in better performance. For example, lowering it from 40 to 30 frames would eliminate almost all false positives without increasing the number of false negatives.

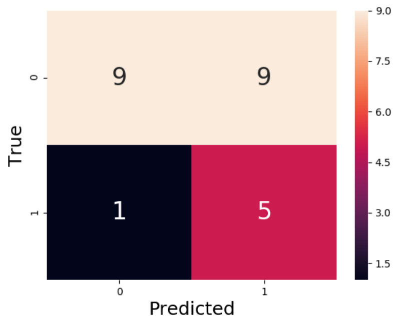


Figure 18: Confusion matrix for per-video classification, using a threshold of 40 consecutive frames, using three different models.

5.2.2 Relation network

The first step in evaluating the relational network model is to monitor the validation metrics during training of the model, namely validation accuracy and validation loss. This is not the same as evaluating on a test set completely disjunct from the training set, but it does give a measure of the performance one can expect from the model. This was done both for the case of encoding video frames with the use of a pre-trained net, as well as encoding them with an autoencoder.

5.2.2.1 Evaluation on Moving MNIST

As the setup for the experiments run on the moving MNIST dataset using the relational network approach when compared to the OCSVM, the results have to be considered carefully when making any comparisons. In order to avoid creating overly long sequences containing mostly redundant information, not every consecutive frame in the sequences was used. Instead, every third frame was selected from a sequence and put together as time steps to be fed into the recurrent network. With the sequences being generated for the purposes of this experiment, the number of frames in each generated video where the same, leading to the avoidance of having to omit short videos, cut long videos off, or introduce some sort of time step padding.

As can be seen from figure 20, the accuracy with which the model could distinguish between similar and dissimilar sequences was slightly above 90%. This is a bit worse than what one could hope for, especially when having in mind the performance of the OCSVM model, in particular its performance using the UCF101 dataset. The reason for the performance not being higher than this might be explained by the data itself. In all of the moving MNIST sequences, there are a number of frames where the digits overlap. In many of these frames, it is difficult even for a human to distinguish between them. An example of such frames can be seen in 21.

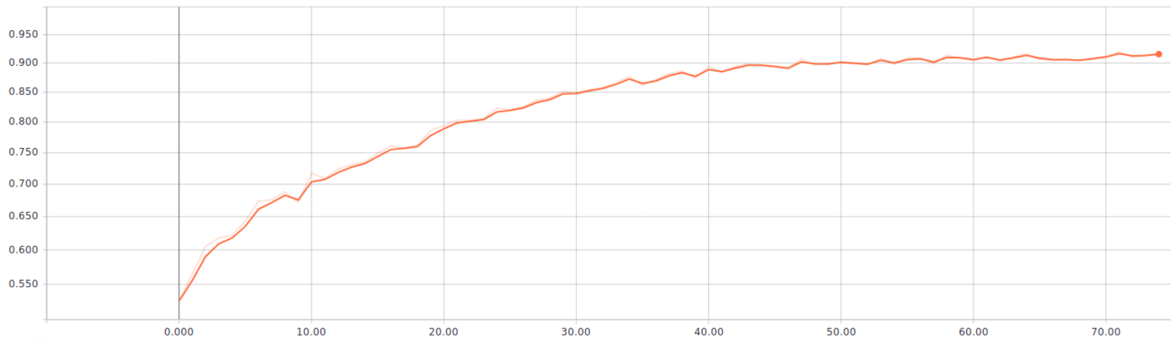


Figure 19: Validation accuracy for the relational net model, trained on the moving MNIST dataset.

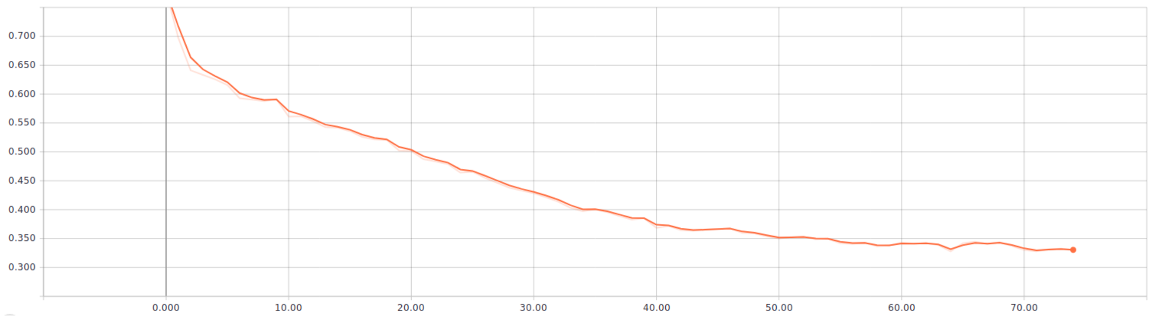


Figure 20: Validation loss for the relational net model, trained on the moving MNIST dataset.

When looking at the performance of the model when testing it on a random sample of moving MNIST sequences, it becomes apparent that the model has a tendency to over fit. Perhaps this is not that surprising, considering the data which make up the dataset. When taking into account the number of frames with overlapping digits and contemplating the performance displayed by the model, it is clear that the viability of the tested method can not be established by tests on this synthetic dataset alone, and further testing is needed.

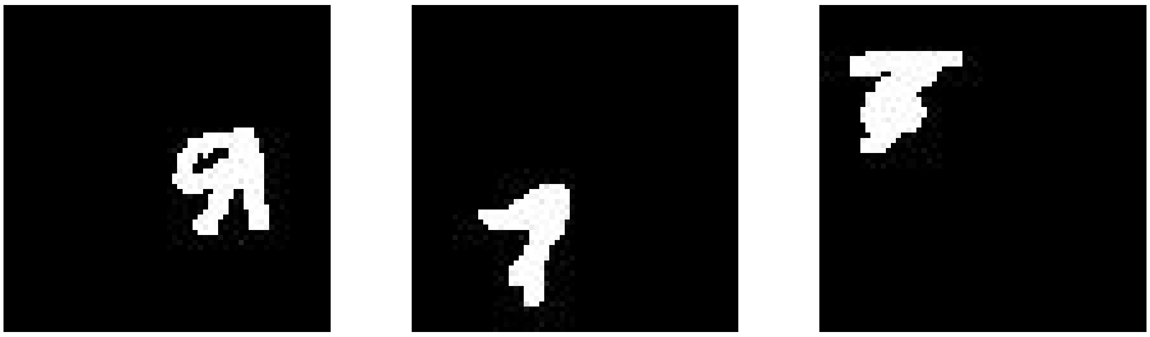


Figure 21: Images from three different sequences, one with a 1 and a 9, one with a 7 and a 9 and one with a 3 and a 5, all with overlapping digits, making it difficult to distinguish between them.

5.2.2.2 Evaluation on UCF101 sequences

Just as is the case with data for the moving MNIST dataset, the results from training the relational network on UCF101 data have to be carefully considered when comparing them with other results, such as those of the OCSVM approach. The reasoning is the same; the accuracy here is not measured on a frame-by-frame basis, but rather for sequences in their entirety. Or to be specific, in the entirety that was selected using the frame skipping method mentioned in section 5.2.2.1.

In addition to the frame skipping method, the maximum number of time steps was also set, in the interest of not having to train the model in several different stages adapted to different sequence lengths. This resulted in the omission of sequences shorter than a certain number of frames, in this case 90. Longer sequences were simply cut off after the appropriate number of frames were extracted. The results of the training can be seen in figure 24 and 25.

Further testing was done on a random sample of a 1000 pairs of video sequences from the UCF101 dataset, with the results as seen in figure 23.The dataset used for evaluation is by no means exhaustive, and the results of evaluation done with this data should not be interpreted as the capabilities of a state-of-the-art system. Rather, it servers the purpose of giving an idea of the feasibility of the approach. It is also important to note that the dataset hasn't been subject to manual pruning in order to remove unfit pairs of sequences like those discovered during the testing of the OCSVM approach. It is also important to consider what exactly is used as correct and incorrect when calculating the performance metrics. Since the model assigns a score in the range of 0 to 1, signifying the similarity between two sequences, a correctly inferred sequence pair is a pair which is similar and receives a score of ≥ 0.5 or a sequence pair that is dissimilar and receives a score of < 0.5.
During testing, the model made some interesting mistakes, image frame examples from which can be seen in figure 26.

5.2.2.3 Use case evaluation

When considering results from doing the smaller scale use case evaluation on footage from the bike room, it should be noted that the model isn't retrained for this purpose. Instead, what is actually evaluated is the performance relative to the performance on the test set from the UCF101 dataset on which the model was trained. This serves the purpose of examining how well the idea of pre-training a distance metric learning network could act on a new type of scene. The tests show that the scenes recorded in the bike room are all much more similar to each other when compared to the UCF101 datasets dissimilar scenes, even from the same category. This results in a clear bias towards classifying such scenes as similar when they are in fact dissimilar. This bias is so extreme that the model classifies every video recorded in the bike room as being similar to all other videos, giving all possible pairs very high similarity scores **??**.

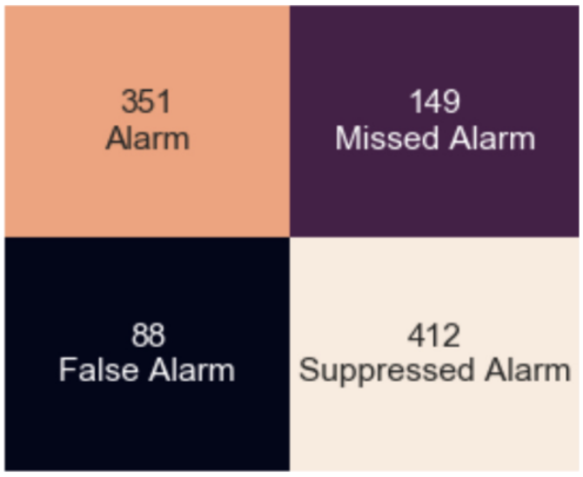


Figure 22: Confusion matrix of 1000 pairs of videos with equal amounts of similar and dissimilar pairs.

Precision:	0.734
Recall:	0.824
F_1 score:	0.776

Figure 23: Precision, recall and F1 metrics for the relation net model on pairs of videos from the UCF101 dataset.

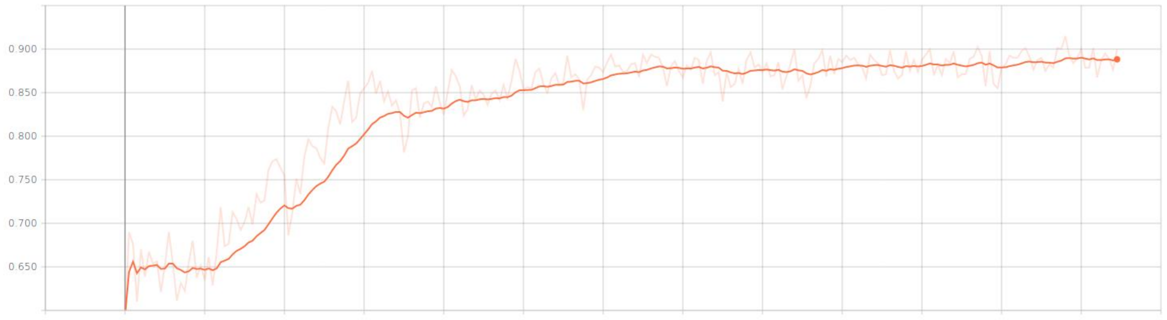


Figure 24: Validation accuracy for the relational net model, trained on the UCF101 dataset.

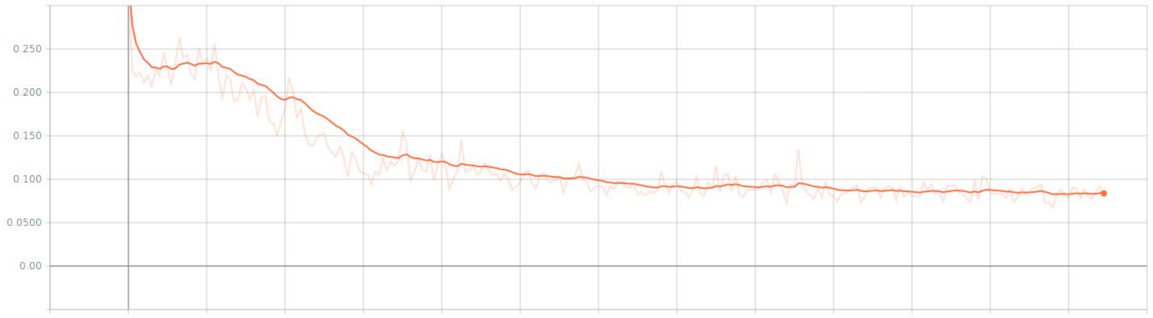
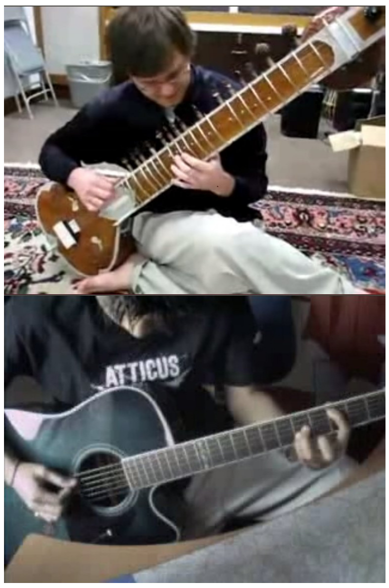


Figure 25: Validation loss for the relational net model, trained on the UCF101 dataset.

(a) Videos from the same scene, receiving a low similarity score. This could be based on the fact that the examples contain different gymnasts, in red and white clothing respectively.



(b) Videos from the different scenes, receiving a high similarity score. Possibly because *what* the scenes depict is similar.

Figure 26: Interesting errors on test sequences with the relational network.

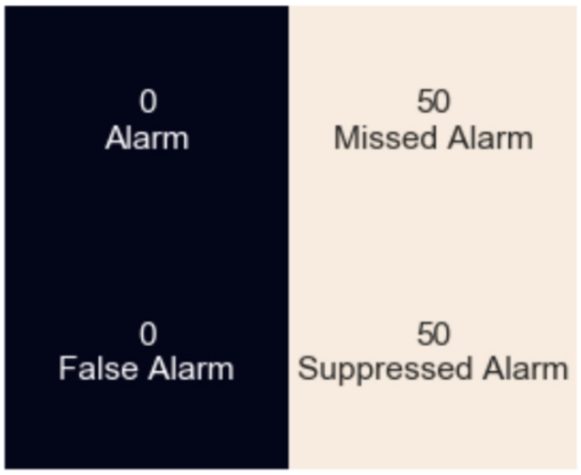


Figure 27: The relational network deems all pairs of video sequences from the bikeroom to be more similar than dissimilar. A clear case of the measure of similarity learned from the large dataset not translating well to this new domain.

6 Discussion & Conclusions

There are two main lines of reasoning regarding the results, namely, questions regarding the general problem, such as what challenges are encountered when using feedback from a user as the basis for filtering video sequences, and more technical aspects of the problem such as the use of transfer learning and using features extracted with a network trained on still images on video sequences. This section will first discuss the more technical aspects, before discussing the challenges around the general problem formulation.

The technicalities regarding the solving of a one-class classification problem are rather involved and lacking a clear-cut, general approach beside the use of more traditional tools such as support vector machines, like the OCSVM classifier used in this thesis. This thesis has discussed the problems of trying to solve a one-class problem with a traditional neural network approach, as a network geared towards classification will learn a trivial solution if presented with only examples from a single class. The use of a multi-class classifier and studying the output distributions in order to infer the membership of a target class of a sample in a one-class classification context was discussed, and while it may produce promising results given some specific setups, it does not generalize very well. What was seen in the results from the use of an OCSVM on features extracted frame-by-frame was somewhat surprising. In many cases the performance was very high, especially when allowing for a number of misclassified frames before classifying a video as a whole. It does however become clear that there are some real limitations with such a solution, something which could be seen on the small test set of recordings from the bike room, where a larger proportion of frames were misclassified when compared to the UCF101 dataset. From the experiment results, both on the moving MNIST dataset as well as the UCF101 dataset, the effects of the quality of the data used for evaluation also was apparent. In the case of the UCF101 dataset, examples were given showing how dissimilar scenes which, based upon their names, should be similar actually were. The examples showing how digits in the moving MNIST dataset overlap in many of the frames also point out a problem with using that particular generated dataset as anything more than a very general indication of performance of a system. The low amount of information in the moving MNIST dataset is probably responsible for the fact that the OCSVM model actually performed better on the UCF101 dataset, which should intuitively be more complex and more difficult to learn a tight enough decision boundary for.

As for the use of transfer learning, which has been employed heavily in this thesis, the results are in line with what has been discussed and shown in previous work, such as [12]. When it comes to the question of how well this method translates to the domain of video, wherein the features are extracted from a single frame at a time, much like the traditional use on still images, a lot of promise has been shown here. The results are especially striking when considering the performance of the OCSVM method on a frame-by-frame basis, which often manages to classify a high proportion of truly positive frames as such, while generally keeping the rate of false positives low. This would suggest that the temporal nature of the sequences is less important for distinguishing between sequences, than what would seem to be the case at first glance.

In an effort to find a way to solve the problem of recognizing video sequences from only positive examples, a distance metric learning relational network was tested. As for the results from these tests, a lot of interesting conclusions can be drawn. The performance of such a network is not only dependent on the quantity and quality of the data, but also on the design of the network itself, as well as the parameters selected during training such as the learning rate. To find an optimal design of the network is extremely complicated, as there is no real established methodology for this. An increase in capacity of the network is closely correlated to the maximum accuracy and minimum loss on the training data, but does not generally lead to better performance on the validation dataset. This is a symptom of over-fitting, and follows the reasoning in [9], where it is stated that over-fitting can occur when some hyperparameter has a large value, in this case the number of nodes. According to the universal approximation theorem, it should be possible to achieve any accuracy required, given that a network is designed large enough[9]. This tendency to over-fit and thus perform worse on the validation data limits the improvement that can be made by increasing the capacity of the network. This tendency to over-fit could clearly be seen when training the model on pairs of videos from the UCF101 dataset and then testing it on pairs of videos recorded in the bikeroom at the Axis offices, where the model deemed all pairs as similar. This limitation is especially crucial when the model is to be used in a new setting, where performance on validation data from the dataset used to train the model might not be indicative of performance in the new setting. The clear way in which the model produces an inferred relation score can however be used to, if not increase performance, clearly show for two sequences if they are deemed similar and not. Overall, considering the limited amount of fine tuning the details of the network architecture and the lack of filtering out unsuitable data, the relational network performs quite well in many cases. Just as is the case with the OCSVM though, there are some sequence pairs which causes the model to behave unpredictably, and it is not always clear exactly why this is the case. This leads to the same problem as for the OCSVM, where it is hard to know to what degree a user can trust such a system to behave in the expected manner.

It is not obvious that one method is clearly superior, but there are a few things which speak in the favor of the relation net approach, even though the performance was terrible when using videos from a completely different dataset than what the model was trained on. The main advantage is the ability to train on an arbitrarily large dataset in advance, hopefully resulting in a learned distance metric that is meaningful in as wide a range of use cases as possible. This could possible make the system less dependent on the quality of the data a user provides it with, and also enable clearer communication with the user of what the system does. For example, for a given video sequence provided from a user, the system could give a similarity score for new sequences immediately, providing the user with information about how the system will act. Tests on pairs of videos completely disjunct from the training dataset does however show that the approach of training in advance on other data is a double edged sword, and the dataset selected for training would need to be carefully examined for problems such as those described in this report, with videos from the UCF101 dataset being labeled in a way so as to suggest them being similar, while in reality this was not always the case.

As for the general question of the viability of using user feedback in the form of examples to have a system recognize similar sequences, it is still an open question. The ambiguities inherent in such an approach might make it unsuitable in many cases. This report has previously touched upon the issue of what is really meant by an motion event, and that question is key to this problem. It must be possible for a user to know with a high degree of certainty what exactly the system thinks it should exclude, given a certain user feedback. As this is hard to know before the system is tested on a wide variety of scenarios for each possible use case, not much can be said about how something along those terms would be achieved. In a surveillance system this poses a major problem, since such a system needs to act in a predictable and expected manner. An example that previously was given is a scenario in which a garbage truck passing through the surveilled area is to be recognized and excluded from events that are to raise a motion alarm. The reasoning that was presented was that there are a lot of ways to interpret such a scenario. Should the same truck driving through the scene in different directions be seen as the same or different events? How similar does another garbage truck have to be to be considered as the same type of event? What about the relationship with other entities in the scene? All of these questions, and more, would have to answered in a clear and consistent manner in order to correctly define what a system should and should not recognize. And even with answers in place for those questions, still more arise when the problem is observed in further detail. How can a user of such a system know what the system *actually* has learned to recognize? The same would be the case even with the use of a human operator. The reader might once again consider the garbage truck scenario, and imagine that a human operator should be given instructions of ignoring such an event, not raising an alarm should it occur. What information would the operator need to accomplish this task? If the operator is presented with a video sequence containing said garbage truck, will the operator truly recognize the event of the garbage truck driving through the scene as what is to be ignored? It could just as well be that the operator interprets this as vehicles in general should be ignored, or it might be something completely different taking place in the scene. The human operator has one great advantage: the ability to ask for further clarification, as well as being asked what has been understood about the scene. In the case of a system being trained using example data, the way what the system has understood about the scene can only manifest itself in the reaction of the system upon encountering another scene. Thus the only way to figure out what the system will do in a given scenario, is to present it with that scenario. This limits the ability to trust the system to act in accordance the the users wishes in new, previously not encountered scenarios. These scenarios might very well be when a surveillance systems operating characteristics are most critical, but it will also be when they are the least reliable.

There is a famous, and probably untrue, story of a machine learning project commissioned by the US army which had the goal of being able to automatically detect tanks among trees in images. A system based on a neural network was designed and trained on images of tanks among trees and tress without tanks among them. On the test set taken from the same dataset the system performed admirably, but as soon as the performance was evaluated in another setting, it performed no better than chance. It turned out that the dataset was such that all the images containing tanks were taken on a cloudy day, and all images of just trees were taken on a sunny day, and thus the system had learned to classify cloudy versus sunny weather. While the veracity of this story is questionable at best, its morals still hold true. It is very important that training of a system is done on data that really represents the problem to be solved. In the case of using as little data as possible, and at that, data supplied by a user as feedback to the system, it is hardly certain that this data will be of the quality required for the system to learn the appropriate things. And just as in the case of the story of the tanks in the trees, a system may very well exhibit good performance on a test set of the same type as the training data, tricking a user into the false belief that the system is doing what the user expects it to do, for it to fail completely given another testing environment.

6.1 Conclusions

This report has thoroughly established the challenges with the problem of using feedback from a user to create a system for filtering motion events in video, both in terms of the general problem and the more technical aspects. From the experiment results presented here, it is clear that working with a low amount of data from a single class is challenging indeed, and especially so in the case of that data being made up of video sequences. Even with the challenges and problems inherit in the problem, a surprisingly good performance was shown with the use of transfer learning for extracting features from individual video frames and using a standard OCSVM model to train on these features. There were however some clear limitations to this approach. Some of these limitations include an erratic behavior under certain circumstances, which can be hard to predict in advance, as shown by the very low performance in some tests. The use of an extra mechanism for classifying a video sequence as a whole based upon a threshold number of consecutive frames classified as either part of the target class or not shows that the pretty basic approach of using a support vector machine actually can deal with sequential data.

As for the transfer learning itself, it proved very successful together with the OCSVM model, and the results from the distance metric learning relation net strengthen the notion that a network pre-trained in another domain can provide meaningful abstractions even in a sequential setting. If another encoding could produce even better representations of the data is still an open question, however.

With the use of a distance metric learning approach, the hope was to avoid the problem of only being able to train on the few positive examples available for a given target class, and instead learn to compare sequences. The hope was that the increased amount of possible training data, combined with the fact that such a model also took the temporal nature of the data into account, being implemented in a recurrent manner, would increase the performance. It was shown that this was not as easy as first thought, mainly because of the data available. In the case of the generated moving MNIST dataset, many sequences deemed dissimilar were in fact very similar to the human eye, a result of the digits which make up the sequences overlapping in many of the frames. In the case of the UCF101 dataset, many sequences deemed similar based upon the naming conventions of the dataset were in fact dissimilar to the human eye, as could be seen in the example given in figure 15. There also seem to be a large difference in what becomes the learned definition of similar in dissimilar on both the UCF101 and the moving MNIST dataset, when comparing the results on these datasets with results on videos recorded in the bike room, where all videos are classified as similar. This complete failure to identify dissimilar sequences is not that strange when considering the data that the system has been trained on. It is however interesting to consider the fact that the more basic OCSVM model is less susceptible to this problem of training data being too different from an actual use case, as it is trained anew for every scenario. Since the OCSVM approach and relation network work in such different ways, it is in the end hard to give a definitive answer as to which performs more favorably.

Many of the challenges of the general problem are not related to technical aspects, but rather to more *soft* aspects, such as how exactly terms are defined, and how concepts are communicated with a user. In the introduction and the discussion sections these problems were discussed and though many interesting things can be said about them, clear answers are still missing. Further studies will be needed to find a suitable way of defining terms like *motion event* in a way such that both a user and a developer understand exactly what is meant. Exactly how a system would communicate its *understanding* of a users requests also needs further exploration. What is certain is the importance of these problems, and it is clear that without clear definitions it is impossible to implement a system of this type that will work in satisfactory manner, no matter the performance of a system.

6.2 Summary

In this report the problem of basing the filtering of motion detection alarms on static filters has been presented, and the need for a more adaptive way to suppress unwanted alarms has been discussed. The approach for solving such a problem that has been discussed is the use of feedback from a user in the form of video sequence examples. The report has discussed two different ways to use video sequence examples to recognize new, previously unseen video sequences and classify them as either similar or dissimilar to the provided examples. The use of transfer learning was discussed and implemented using a neural network pre-trained on the ImageNet dataset, which extracted features from image frames from the video sequences. Both methods were tested on a generated dataset, made up sequences of moving digits from the MNIST dataset, as well as videos from the UCF101 dataset and additionally on a small dataset of videos recorded for the purposes of this thesis. The results showed that a basic system using OCSVM can perform quite well when working on features extracted with a pre trained network, even though the dimensionality of such features were quite large. It also became apparent when studying the performance of a OCSVM model on the standard MNIST dataset that feature extraction is not always beneficial, which was the case with images from the MNIST dataset being made up of only $28 \cdot 28$ pixel images.

The effects of the quality of a dataset has also been explored, and the report showed the detrimental effects of relying solely on the conventions used for naming different scenes in the UCF101 dataset, as this was not always a good basis for the distinction between similar and dissimilar sequences. The negative effects of the problems with the dataset were especially apparent in the case of the relational network, as it relied on being able to train on a large number of video sequence pairs in a semi-supervised manner, where the label of the pair, being either similar or dissimilar, was dependent on the naming of the video sequences. The effects of this could also be seen in the case of the OCSVM approach, though to a lesser extent as the problems were naturally limited to models trained on scenes with a high intraclass variance.

The OCSVM methods reliance on a new model for each scenario has both benefits and drawbacks. As discussed before, this could help make it less susceptible to limitations in the data for a given scene, as the effects would be limited to the model trained on that scene. At the same time, the requirement to train a new model for each possible scenario does not scale very well. Each time a user would like to exclude a new type of motion event, a new model would have to be trained. While the training of a OCSVM model is far less computationally expensive than a neural network, it is still a step that needs to be taken, and the models though lightweight still require some storage space. There is also the problem of scale when it is time to infer the class of a new video sequence, since every model would need to be run with the sequence as input before a definitive decision on the class affiliation of the sequence.

The use of a relational network solves many of the shortcomings of the OCSVM approach, in that it could be trained on a large, well-crafted dataset before it is used by a user. The user would then need only to supply example videos, with which new video sequences would be paired and their similarity calculated. This does however introduce new problems, mainly regarding the time and resources needed to ensure a large and representative enough dataset is used. The UCF101 dataset used in this report seemed unfit for this task, as the relation network model performed well on a test set from this dataset, but could not distinguish between the sequences recorded in the bike room at the Axis offices. The discussion section included a discussion of the problem of knowing what the model actually has learned, which is a very difficult problem, and puts a limit on the reliability of the system.

The discussion section also discussed the problems with the general idea of using examples of events as a basis for filtering new events, classifying them based on the idea that they are either similar or dissimilar to events that a user want to ignore, and suppress motion alarms from. The challenges around this discussed mainly revolve around the problem of defining exactly what makes up an event, and communicating between the system and the user in a way such that confusion around this can be avoided.

6.3 Future work

Though this thesis has shown that there are numerous problems with a feedback based recognition system, there are several interesting avenues for future work. One such promising approach is the combination of a neural network with a support vector machine for one class classification. By training a recurring network with the goal of minimizing an enclosing *hypersphere*, and using this as the loss function, a one-class decision boundary can be established [27]. A similar approach is discussed in [28], where the authors use a feed-forward network to achieve the same goal, rather than a recurrent network. By incorporating the use of a OC-SVM into the training of the network, the network can be able to learn a better embedding when compared to just using a OC-SVM on features extracted with a pre-trained network, as descriptive as such features might very well be. Learning representations that are expressly tailored to the use case at hand has been discussed in various papers, such as [29], and has shown great promise when it comes to solving the problem of working with very limited data. Possibly such an approach may work even in the recurrent, sequence based case, forcing a network to learn a representation that is meaningful in the sense that it enables the intra-class distances to become small for an entire video sequence and not just still images. This way, it could be possible to learn a representation that is *good* in the sense discussed in [9], that is, enabling the separation of underlying causal factors.

Another possibility to enhance the performance of a system aiming to learn video sequence similarity and dissimilarity would be to improve the data available, both in quality and quantity. This could help alleviate some of the problems with sequences that according to the naming convention of the UCF101 should be similar, but in reality are very dissimilar. To exclude scenes with a lot of variance between examples that based on their name should be similar could benefit the model performance wise. A problem with this is that manual pruning of unfit data is very time consuming, as it requires manually inspecting each video sequence, and in the case of UCF101 they number 13320, making for a very ambitious task. It is also hard to imagine a way to automate such a process, as this would require a system to be able to do what the system the data is needed for is supposed to do. Another possible area of experimentation related to the dataset could be the use of generated or synthetically augmented datasets to increase the coverage of available data.

During this thesis, the whole frame of the video sequences has been used. A possible future area of experimentation would be to use cropped video sequences, which only display what gave rise to detected motion, for example by putting a bounding box around a moving object and cropping the image frames to this bounding box. With such an approach, it would be possible to distinguish between multiple concurrent moving objects. This would also limit the possibility that a system learns the wrong thing, as exclusion of all objects not being the one causing the motion the user wants recognized lets the system focus on only that, and lets the user be sure of what it is the system is actually learning from.

References

[1] Henrik Lindelöf Bilski. Heterogeneous sensor fusion: Verification and optimization, 2017. Student Paper.

[2] Jonathan Lundholm, Bibby Steneram, Paul Maxwell. False alarm filtering within camera surveillance using an external object classification service, 2017. Student Paper.

[3] Ronald Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

[4] Shehroz S. Khan and Michael G. Madden. A survey of recent trends in one class classification. July 2017.

[5] David MJ Tax, Robert PW Duin. Support vector data description. *Machine Learning*, pages 45–66, 2004.

[6] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and Shah R. Signature verification using a "siamese" time delay neural network. 1994.

[7] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. Learning to Compare: Relation Network for Few-Shot Learning. *ArXiv e-prints*, November 2017.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *ArXiv e-prints*, December 2015.

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[10] Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves. Playing atari with deep reinforcement learning. 2013. https://arxiv.org/pdf/1312.5602v1.pdf.

[11] Grant Sanderson. 3blue1brown. Available at `https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi`.

[12] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. 2014. https://arxiv.org/pdf/1403.6382.pdf.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv e-prints*, April 2017.

[14] Imagenet image dataset. available at `http://image-net.org/`.

[15] Keras. Available at `https://keras.io/`.

[16] Tensorflow. Available at `https://www.tensorflow.org/`.

[17] Python programming language.

[18] van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, Gouillart E, Yu T. scikit-image: image processing in python, 2014. Available at `http://scikit-image.org/`.

[19] scikit video. Available at `http://www.scikit-video.org/stable/io.html`.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Y. Bengio Y. LeCun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11):2278-2324*, November 1998.

[22] moving mnist. Available at `https://gist.github.com/tencia/afb129122a64bde3bd0c`.

[23] Khurram Soomro, Amir Roshan Zamir, Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild.* Center for Research in Computer Vision, Orlando, FL 32816, USA, 2012.

[24] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *ArXiv e-prints*, March 2015.

[25] Tensorboard. see `https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard`.

[26] Sixin Zhang Yann LeCun Rob Fergus Li Wan, Matthew Zeiler. Regularization of Neural Network using DropConnect. 2013.

[27] A. F. Agarap. A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data. *ArXiv e-prints*, September 2017.

[28] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[29] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, S. Pankanti, R. Feris, A. Kumar, R. Giryes, and A. M. Bronstein. RepMet: Representative-based metric learning for classification and one-shot object detection. *ArXiv e-prints*, June 2018.