Video Traffic Classification

A Machine Learning approach with Packet Based Features using Support Vector Machine

Videotrafikklassificering
En Maskininlärningslösning med Paketbasereade Features och Supportvektormaskin

Simon Westlinder

Faculty of Health, Science and Technology

Degree Project for Master of Science in Engineering

30 Hp

Supervisor: Johan Garcia

Examiner: Donald F. Ross

2016-06-07

Abstract

Internet traffic classification is an important field which several stakeholders are dependent on for a number of different reasons. Internet Service Providers (ISPs) and network operators benefit from knowing what type of traffic that propagates over their network in order to correctly treat different applications. Today Deep Packet Inspection (DPI) and port based classification are two of the more commonly used methods in order to classify Internet traffic. However, both of these techniques fail when the traffic is encrypted. This study explores a third method, classifying Internet traffic by machine learning in which the classification is realized by looking at Internet traffic flow characteristics instead of actual payloads. Machine learning can solve the inherent limitations that DPI and port based classification suffers from. In this study the Internet traffic is divided into two classes of interest: Video and Other. There exist several machine learning methods for classification, and this study focuses on Support Vector Machine (SVM) to classify traffic. Several traffic characteristics are extracted, such as individual payload sizes and the longest consecutive run of payload packets in the downward direction. Several experiments using different approaches are conducted and the achieved results show that overall accuracies above 90% are achievable.

Keywords: Supervised Machine Learning, SVM, Video traffic classification

Acknowledgements

I would like to thank Procera Network for the opportunity to work with such an interesting subject. Special thanks to Anders Waldenborg who was the supervisor at Procera and who also provided the data that was used in this study. And finally, I would like to thank Johan Garcia who was my supervisor at Karlstad University for the help with my dissertation and also for the help he provided beside the dissertation.

Contents

List of figures

List of tables

1. Introduction

The number of Internet users' increases yearly and today about 40% of the population has an Internet connection [**1**]. Internet service providers (ISPs), governmental and private organizations are highly dependent on Internet traffic classification and it can be used for a number of different reasons [**2**]. ISPs and network operators can by traffic classification solve different network management problems. ISPs and network operators need to know what type of traffic that propagates over their network in real time in order to give the best and correct treatments to the traffic in order to fulfill their various business goals, e.g. keeping their customers happy. Internet traffic classification can also be used in automated intrusion detection systems by finding anomalies in the traffic patterns that may indicate some sort of attack.

Two of the more conventional methods used for classifying Internet traffic is by Deep Packet Inspection (DPI), which means that the packets payloads are directly inspected, in addition to classifying the traffic by well-known port numbers [**3**]. These two techniques are also known as payload based classification and port based classification respectively. However, as discussed in [**4**] and [**3**] these techniques have some inherent limitations: (i) computational complexity, (ii) privacy issues, and (iii) the incapability to handle the increasing usage of encryption and obfuscation techniques. As an example of this incapability, applications may encrypt the data that are to be transmitted in order to avoid being detected. Applications may also use dynamic port numbers instead of well-known ports assigned in the Internet Assigned Numbers Authority (IANA) registry [**5**], which as a result leads to a greater difficulty of identifying and classifying the network traffic.

Procera Network is a business company that offers several solutions for network operators and vendors to monitor, manage and monetize their network traffic in order to, mainly improve customer/subscriber experience, but also to distribute their Internet links in a more effective and profitable manner [**6**]. These solutions are realized by classifying the Internet traffic in order to be able to give the proper handling and treatment to respective application. Today Procera classifies Internet traffic by mainly Deep Packet Inspection (DPI), which has been and still is one of the more accurate techniques used. However as previously mentioned DPI has some inherent limitations and as more and more traffic applications uses encryption or various obfuscation techniques classification by DPI becomes insufficient.

A third method that can be used to classify Internet traffic is by Machine Learning (ML) which is the subject of this study. The goal of this study is to explore if ML methods can be used to classify Internet traffic and solve the inherent limitations the two former mentioned methods suffers from. ML-based classification is about inspecting and extracting traffic characteristics instead of using the packets actual payloads. In ML terms traffic characteristics are called features and features such as packet sizes, packets inter arrival times and flow duration for example can be extracted from each individual flow. Each individual traffic flow is then described by the same set of features. In short the process of building a ML classifier is to train a classifier on already collected data where the traffic classes are known in advance. The classifier can then be applied on real time traffic to determine which classes the traffic flows belongs to. The machine learning process is depicted in Figure 1 and the process of training the classifier is executed by tracing the blue path while the black path illustrates the process when predictions on unseen data is to be carried out.

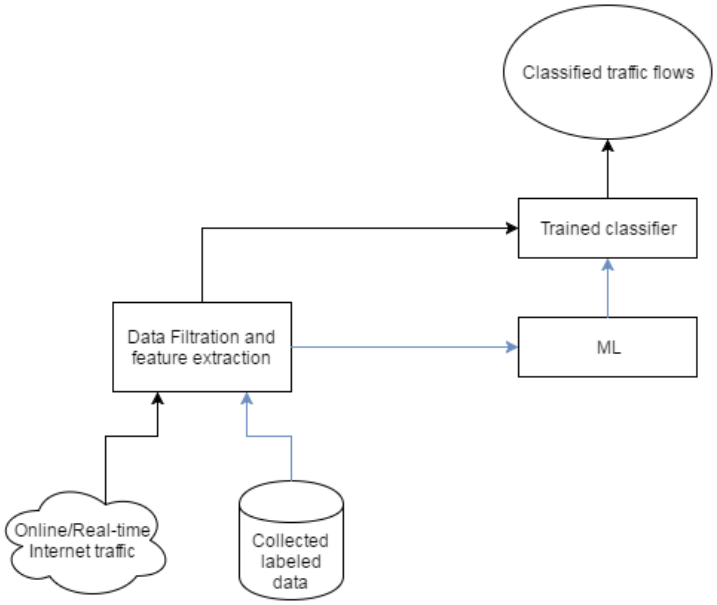

Figure 1 The machine learning process.

In this study the Internet traffic has been divided into two different types of traffic, namely Video and Other. In total the Internet traffic provided consisted of ~375 different services where ~90 of those were considered video traffic. The features used in this study are all based on the first N number of packets. More specifically two different implementations regarding the computations of the features have been used, one in which all features are based on the first N

number of payload packets alone and one where the features are based on the first *N* number of packets. There exist many different ML algorithms and in this study the Support Vector Machine (SVM) algorithm has been used and its performance will be compared to a Decision Tree and a Random Forest algorithm which has been evaluated in a parallel study performed by Henrik Johansson [**7**].

1.1. Goal

The aim of this study is to evaluate different Machine Learning (ML) techniques in order to overcome the previously mentioned limitations introduced by the DPI and port based techniques to make Internet traffic classification possible without the use of the actual data in the packets. Instead of using DPI the traffic classification is to be realized by inspection and extraction of Internet traffic characteristics, which has been suggested by previous works to be effective.

ML methods have been used in order to classify Internet traffic with the use of either packet based features or flow based features. In one of the more recent studies by Lizhi Peng et al. [**8**] promising results has been achieved while only considering the first 6 payload packets and the studies found in [**9**], [**3**] and [**4**] also shows promising results when only considering the first few packets. In a study by Jayeeta Datta et al. [**10**] an attempt at classifying encrypted traffic by ML was carried out which also showed promising results.

However, the DPI and port based techniques can still be used to provide the machine learning methods with ground truth data, i.e. labeled test- and training data. As a starting point, Procera specified that one of the more interesting types of Internet traffic is those that contain video, which implies that as a first milestone video traffic is the type of traffic that should be distinguished and if there is time the traffic can be divided further. Procera clarified that one of the more important requirements of the classifiers is that they should be as fast as possible during the classification of new observations, the time it takes to setup and train the classifier is of less importance.

1.2. Implementation Tools and Resources

The programming language that was used during this study was Python. In this sub-section different tools regarding Python will be briefly presented in addition to a resource for large scale computations that was used.

1.2.1. SNIC

The Swedish National Infrastructure for Computing (SNIC) provides a set of resources for large scale computation and data storage [**11**]. The resources are provided through open application procedures.

1.2.2. Jupyter Notebook

The Jupyter Notebook is an interactive computational interpreter that has support for over 40 programming languages including Python [**12**]. Jupyter is an environment which offers fast testing of code without having to create test files and it offers the ability, together with modules, to display rich data and plots as a result of computations. There are two components involved: The first component is the Jupyter Notebook web application which offers an interactive User Interface (UI) where all code and its outputs are stored in persistent cells, which may be further edited. The second component is plain text documents which are called notebooks. The notebooks are for recording all the computations that are carried out, including all the input and output, and they also serve as an easy way of distributing the written code and the results of the computations.

1.2.3. Pandas

Pandas is an open source library for Python which offers high performance data structures for data manipulation and data analysis [**13**]. Pandas provide the two data structures DataFrame and Series. The former data structure, the DataFrame, was the one mainly used in this study. A DataFrame represents the data in tabular form, with rows and columns, and it also offers a number of inbuilt functions that allows manipulation of the data in a fast and easy manner.

1.2.4. NumPy

NumPy is an open source library for Python which offers tools for scientific computations [**14**]. NumPy provides a powerful data structure, the n-dimensional array, also known as the NumPy array, together with high-level mathematical functions that can operate on NumPy arrays. When working with data of a great magnitude the performance of operations done on the data may be critical and this is where NumPy arrays can be a great asset because NumPy arrays offer high performance operations when working with big data sets.

1.2.5. Scikit-learn

Scikit-learn is an open source library for Python which provides tools for machine learning [**15**]. It offers a number of machine learning algorithms for classification, regression and clustering. It also includes different algorithms for feature selection and different tools for evaluation of the models.

1.3. Disposition

In Chapter 2 an overview of machine learning will be presented and the SVM method will be discussed in more detail. In addition the Decision Tree and Random Forest methods will be briefly discussed as well as some other methods.

In Chapter 3 a number of similar studies will be discussed, including their most important results together with which machine learning methods that were used including which features were considered.

Chapter 4 goes into more detail of how this study was carried out. The data set together with the sampled data sets will be presented and the different implementations of the feature sets will be discussed. The evaluation metrics that were used during this study will be presented in addition to some techniques that were used.

Chapter 5 goes through the Scikit-learn tools that were used, including the SVM machine implementation of the ML algorithms.

Chapter 6 presents detailed results for 1 of the 6 experiments conducted in this study together with brief summaries of the additional experiments.

Chapter 7 concludes the study and also provides a brief discussion regarding what may be interesting to explore in future studies.

2. Machine Learning

In this chapter machine learning will be discussed with an orientation towards supervised machine learning. The support vector machine method will be discussed in detail followed by brief discussion of some other methods, such as Decision Tree and Random Forest.

2.1. Overview

Machine Learning can be utilized in a wide range of application areas. It can be used for data mining, image recognition, fraud detection and medical diagnosis for example. Machine learning is a field that is quite big and it is rapidly expanding; it is continually being divided into subfields which are oriented towards their own specialties and types of machine learning. However, this report will not give an extensive coverage of machine learning. There are already good literature that does this [**16**] [**17**]. Instead, in this section machine learning will be introduced and discussed to give a basic overview with some of the more important details highlighted. In short one can say that machine learning is all about learning a solution to a problem from some sample data.

While there is no established definition for machine learning, there are some good statements made that capture the essence of what machine learning is. According to [**18**] in 1959 Arthur Samuel said that "Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed".

In 1997 Tom Mitchell gave a more descriptive statement of what machine learning is. He defined it as: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E" [**19**].

Let us look at a simple example where we apply the latter definition: In an e-mail system the users typically want to be able to filter the e-mails, so that spam and non-spam arrive in different folders. A machine learning system could be used to filter the e-mails (task T) by training the system on already received e-mails (experience E) in order to better discriminate future e-mails (performance measure P).

As mentioned before, machine learning can be divided into different subfields. However, there are two distinct groups which divide machine learning techniques: Supervised machine learning and unsupervised machine learning. In the former group, the system is trained on already known data, i.e. predefined data which is also known as *labeled data*. The trained system can then be fed new unseen data and reach an accurate conclusion or prediction regarding the new data. In contrary, in the latter group, the system is fed unknown data and all by itself it needs to find patterns and relationships in the data which divides the data into distinctive groups.

Since Procera provided already labeled data, i.e. in the data provided the different Internet services the traffic flows belonged to was known, the technique used in this study was supervised machine learning. Hence the rest of the report will be focused on supervised machine learning.

2.2. Supervised Machine Learning

A supervised machine learning system consists of a training phase in which the system takes labeled data as input and trains a classifier, a trained classifier is commonly referred to as a *model*. The classifier, or model, can then be used to carry out predictions on new unseen data. Since the machine learning models are used to predict new data it is very important to know how good a model is, i.e. how accurate one can expect the model to perform given new data. Depending on what the model predicts and what is expected of the model, different evaluation metrics can be used which is covered in Section 4.2.

Supervised machine learning can further be divided into regression and classification methods; where the former predicts a value along a continuous scale while the latter predicts a value in a discrete set of values [**16**]. As an example, let us consider a system that tells us something about the price of a house with regard to some characteristics, such as location, year of construction, and so on. If the system predicts the price of a house in the form of a numerical value, which lies within a continuous scale, it relates to regression. In contrary, if the system predicts a value of the price, “expensive” e.g., in a predefined set, {“cheap”, “expensive”}, it relates to classification. In this study the goal was to explore whether machine learning could be used to classify Internet traffic flows, which implies that an Internet traffic flow is to be predicted belonging to a certain type or class of Internet traffic, thus this was a classification problem.

2.2.1. Features and Feature Vector

In machine learning a *feature* is a measurable property, attribute, quantity or characteristic in the data being observed [**20**]. If an Internet traffic flow is considered, a feature can be the mean payload size, the mean inter-arrival time or the payload size for a single packet for example. The result from extraction of features from a single Internet traffic flow constitutes a *feature vector*. All of the traffic flows considered is described by the same set of features, constituting a set of feature vectors. The machine learning models take these feature vectors as input and give some output regarding the supplied data.

2.2.2. Feature Engineering

Feature engineering is a vital process in machine learning and it can be seen as the process of transforming raw data into features that describes the raw data in the best possible way which the machine learning algorithms can use in order achieve their best possible performance [**21**]. Feature engineering has two important objects: (i) get the most out of the raw data that is at hand and (ii) present the data to the machine learning algorithms in the best possible manner. Both these objects can improve the performance of the models when used on new data. The feature engineering process will result in the *initial feature set*.

2.2.3. Feature Selection

When all the features are extracted and computed, i.e. the *initial feature set* has been decided, there can be features that do not contribute during the classification process and these features can be removed in order to further improve the performance of the model. There are many methods which can be used in order to decide which features are the best to use and which features to remove. In scikit-learn there are built in feature selection algorithms which can be used. Another approach is to evaluate models using different handpicked features and see which features influence the result in the best way. Which features to keep also depends on which machine learning technique is used.

2.2.4. Bias-Variance tradeoff

When working with machine learning methods there are errors introduced. When discussing prediction models, i.e. supervised machine learning models e.g., the errors can be divided into two main subcomponents: error introduced by *bias* and error introduced by *variance* [**22**]. The sum of the bias and variance constitutes the prediction error for a learned classifier.

The error introduced by bias can be defined as the difference between the expected/the average prediction the model outputs and the correct value which is to be predicted. Imagine that several data sets are used during the training and testing of a model. Then the model is said to be biased for a particular input *x* if the model systematically, i.e. not randomly, outputs incorrect values when predicting the correct output for *x*.

The error introduced by variance can be defined as how much the models prediction varies for a given input. If again several data sets are considered and used during training, the model is said to have high variance if the output value differs when taking a particular value, x, as input.

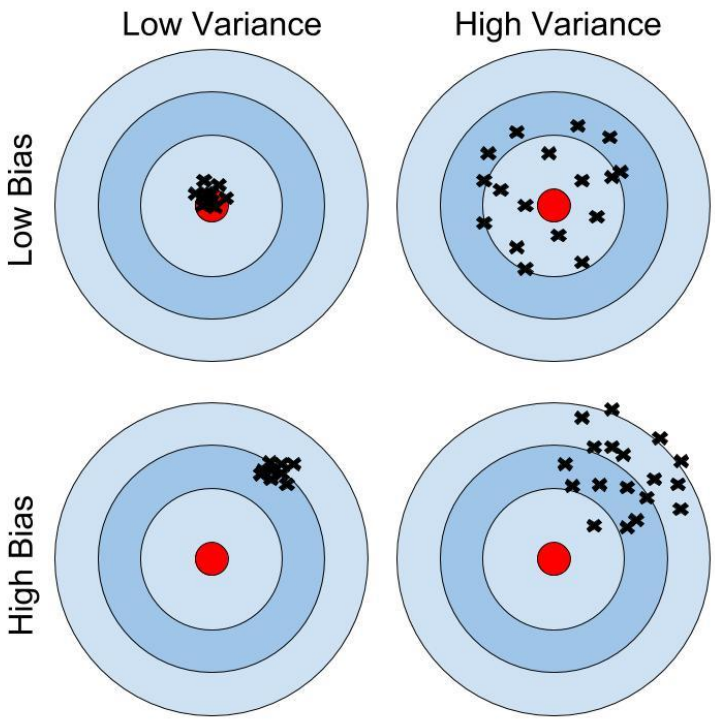

*Figure 2 Bias and variance are illustrated in a bull's eye diagram, adapted from [**22**].*

In Figure 2 bias and variance are illustrated in order to give a better understanding of the error they introduce. The red dot, the bull's eye, represents a perfect model, i.e. a model that perfectly predicts the correct values for all possible data that can be generated and the further out the worse the models get. In order to achieve a perfect model, all possible data that can be generated is needed in the training phase, however, this is generally not achievable. Instead, only a subset of possible data is used during training which yields an approximation of the perfect model. The black crosses are a representation of each of these approximation models. As the figure depicts, the best would be to have as low bias and variance as possible, but unfortunately this is not achievable. There is always a tradeoff which has to be considered: If bias is lowered, the variance is increased, and vice versa. The tuning of a models parameters and which features are used is of uttermost importance if as good a result as possible is to be achieved.

2.2.5. Overfitting and Underfitting

Overfitting and underfitting are highly linked to the bias and variance. If a model is biased, i.e. it has high bias, the model is said to be underfit and if a model has low bias and high variance the model is said to be overfitted [**23**]. Overfitting is for the most part the greater issue in machine learning, which will be discussed in greater detail than underfitting.

A data set mainly consists of two components; the signal, i.e. the data of interest, and the noise, i.e. random fluctuations in the data that has nothing to do with the predictions. However, it can be hard to identify what is signal and what is noise. Overfitting generally occurs when a model is too complex, it can be that the model has too many parameters relative the number of observations e.g. A model that has been overfit will generally give bad predictions when presented with new data, due to the fact that the model captures too much of the noise presented in the training data.

In supervised machine learning overfitting can occur as a result of the training phase. When the model is trained a set of data is used, and the models performance is maximized considering that particular training data. However, the models performance is measured on data that has not been used during the training phase. Overfitting occurs when the model is trained to fit the training data too much; instead of learning from the underlying behavior the model rather memorizes the data.

Underfitting, in contrast to overfitting, is when the model is incapable of capturing the data good enough.

The generalization error is composed of two components; the training error and the validation error. And in order to decide whether a model is overfitting or underfitting the training error and validation error can be evaluated [**23**]. If both the training error and the validation error are high then the model is underfitting, while if the training error is low and the validation error is high, the model is overfitting. There are several techniques used in order to counteract overfitting, and one of them is cross-validation, which is discussed in Section 4.3.

2.3. Supervised Machine Learning Methods

There are many different approaches of supervised machine learning. This sub-section starts with a detailed discussion about Support Vector Machines (SVMs) which is the approach that has been in focus during this study followed by brief presentations of Random Forest and Decision Tree, which were used in the parallel study, and lastly some other common approaches will be presented.

2.3.1. Support Vector Machine

2.3.1.1 The Concept

A SVM is a supervised machine learning method used for classification purposes. A SVM needs training data and is used to predict which class an observation belongs to. Originally SVMs were only designed for binary classification purposes, i.e. the SVMs could only predict whether an observation belonged to either of two classes. However, SVMs have been extended in order to be able to handle multi-class classification problems. There are two main techniques that can be used in order to extend binary SVMs to multi-class classifiers; either (i) directly extend the binary classifier to handle multi-class problems, or (ii) combine multiple binary classifiers into "one" big classifier. In scikit-learn the latter approach has been used and it has been implemented according to two different schemes; One vs. Rest (OvR) and One vs. One (OvO) [**24**]. According to the OvR scheme a single classifier per class is trained, where samples from a single class are chosen to be the positive samples and the rest the negative samples. When this scheme is used the classifier need to produce a confidence score rather than a single class

and when predictions on new data are carried out the class with the highest confidence score are chosen. According to the OvO scheme a problem where there are K classes $K(K-1)/2$ binary classifiers are trained where each classifier receives samples from two classes. When predictions on new data are carried out some voting scheme is applied, e.g. majority voting, between all the combined classifiers and the class that get the most votes gets predicted [**25**].

In short, the main idea behind a SVM is to find the most optimal *separating hyperplane* which maximizes the *margin* of the training data [**26**] [**27**]. A hyperplane is a plane which divides its ambient space into two disconnected spaces, i.e. in an n-dimensional space a hyperplane is of n-1 dimensions. The number of dimensions in a SVM is equal to the number of features used and a SVM can be used for any number of features. However, in this section at most three features will be used, i.e. at most a three dimensional space, in order to keep it simple and understandable. A *separating hyperplane* is simply a hyperplane which divides the observations into distinct groups. The margin, as the name suggests, is a distance in the Euclidian space, which will be explained in more detail later in this section. First off, lets us look at a simple example explaining the main concept behind SVMs.

Assume some data has been collected, i.e. the training data, and the goal is to build a SVM that can predict which one of the two classes *C1* and *C2* unseen data belongs to by looking at the two features *f1* and *f2*, that have been extracted from the data. In Figure 3 the data has been plotted with the features on each axis and as can be seen this is a two-dimensional space. Note that the hyperplane is in fact a line in two dimensions. However, it is known as a "hyperplane" regardless of dimensions. As can be seen the hyperplane clearly separates the data into two distinct groups, which is the goal of a SVM. When predictions on unseen data, i.e. new data, are carried out the observations are predicted to belong to either class depending on which side of the hyperplane the observations belongs to.

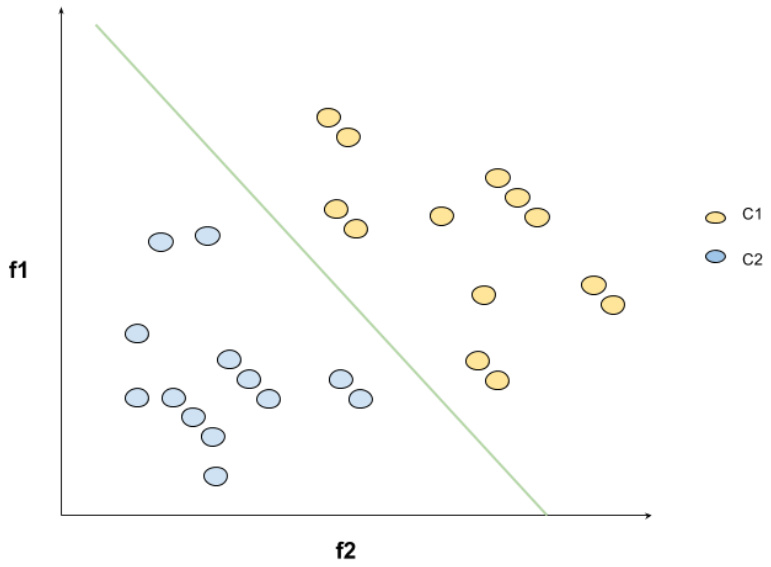

Figure 3 A hyperplane that clearly separates the two classes C1 and C2.

The concept for a SVM is simple to understand; however, as can be seen in Figure 4 a number of separating hyperplanes can be found, actually there can possibly be infinitely many separating hyperplanes. So the problem is to find which hyperplane of all the possible hyperplanes that is the best one, which will be explained shortly. But first, as mentioned earlier, the goal is to find the most optimal separating hyperplane, and this is because that the most optimal separating hyperplane generalizes the best on unseen data. This is easily understood by a simple example.

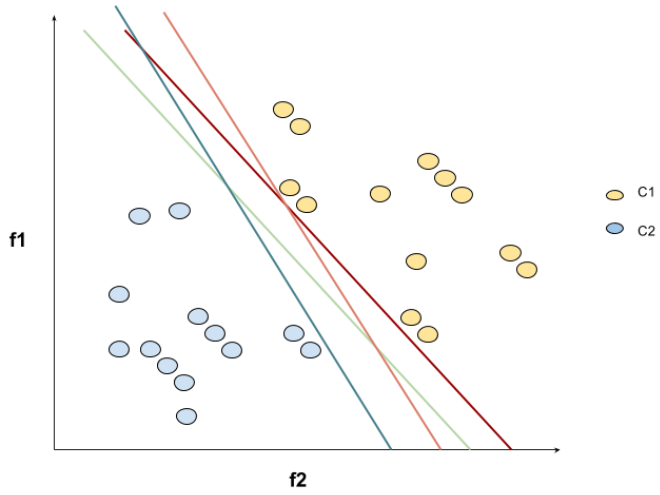

Figure 4 Multiple hyperplanes that separate the data into two distinct groups.

Assume that the red hyperplane in Figure 4 is chosen as the separating hyperplane, i.e. it is chosen as the decision function. According to the red hyperplane the two new observations that belong to the class *C1* are wrongly classified as belonging to class *C2*, as can be seen in Figure 5. This is the result of a poorly chosen separating hyperplane. If a hyperplane, as the red hyperplane, is chosen which is close to observations of one class, it might result in poor predictions when presented with new observations, i.e. the model might not generalize well. A much better choice of a separating hyperplane in this case would have been the green hyperplane, as this would have made perfect classifications given the two new observations. The most optimal hyperplane correctly classifies the training data and when making predictions on new data it generalizes better than other hyperplanes. The problem of finding the most optimal hyperplane is a matter of finding the hyperplane that is as far away as possible from the nearest data points from each class and this problem is solved with the help of margins. Recall that the most optimal hyperplane is the hyperplane that has the largest margin. We will come back to Figure 5 later and see why the red hyperplane is worse, but first let us explain what margins are and how to calculate them.

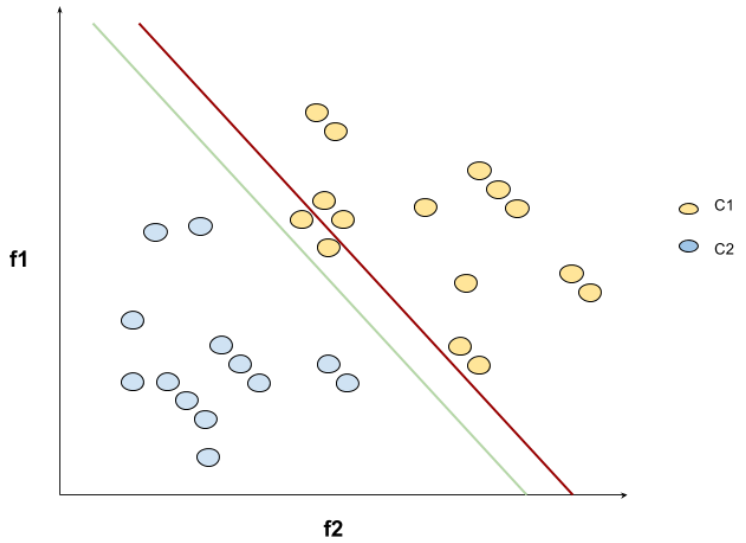

Figure 5 If the red hyperplane is chosen, the two new observations are wrongly classified. If instead the green hyperplane would have been chosen the observations would have been classified correctly.

Margins can be calculated in at least two different ways. The margin for a specific hyperplane can be calculated by doubling the distance to its nearest data point, as seen in Figure 6*a*. Another way of calculating the margin is by taking the distance between two hyperplanes, as seen in Figure 6*b*. If we trace the middle of the margin in Figure 6*b* and draw a perpendicular line, i.e. a hyperplane, we get the green hyperplane as seen in Figure 6*c*. So it can be seen from this that hyperplanes and margins are closely related and an important conclusion can be made: Finding the most optimal hyperplane is the same as finding the largest margin. The largest margin is simply found by selecting two hyperplanes that separates the data with no data points from either class between them and then maximize their distance. Another important concept is *support vectors* which are the data points closest to a hyperplane. The support vectors are the data points that define the hyperplane that is chosen as the decision function and are therefore vital. The decision function is used to classify new observations.

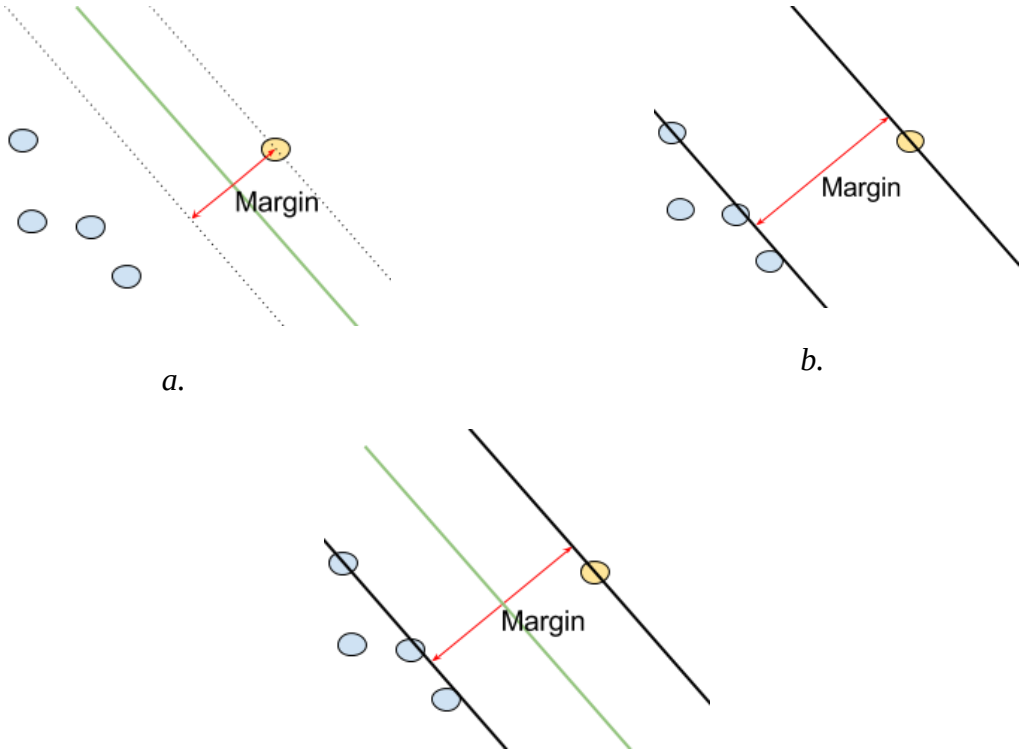

Figure 6 a, b and c. In a. the margin for the green hyperplane is calculated by doubling the distance to the closest data point. In b. the margin is calculated by taking the distance between the two hyperplanes. In c. the green hyperplane is found by tracing the middle of the margin seen in b.

In the discussion about SVMs above only linear separable data has been discussed. However, in reality this is not typically the case. SVMs can be divided into two groups; one that solves the liner case and one that solves the nonlinear case. These two different approaches will be briefly discussed next.

2.3.1.2 Linear SVM

Regarding the linear SVMs there are two types; *hard* margin and *soft* margin SVMs. If the data are completely linear separable, as seen in Figure 3, then a hard margin SVM can be used. However, as previously mentioned, this is usually not the case. If the data are nonlinear separable, as in Figure 7, then a hard margin SVM would fail since a linear separating hyperplane cannot be found. However, as it seems there may be some error or noise in the data and it would suffice to find a hyperplane that separates the data but not completely. This is where the soft margin comes in. A soft margin SVM tolerates some error in the training data and can still find a separating hypeplane, e.g. the green hyperplane in Figure 7. A soft margin SVM is handy where the data may be noisy but in cases where the data are more complex, or even seemingly random structured, even a soft margin SVM may be insufficient.

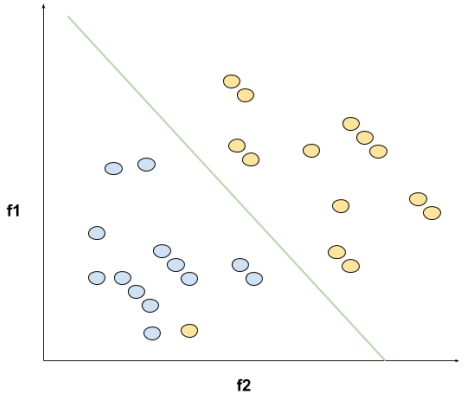

Figure 7 Errors introduced in the training data making it nonlinear separable.

2.3.1.3 Nonlinear SVM

When the data are linear non separable and where a soft margin SVM would not suffice another kind of SVM is needed. The nonlinear SVM can find a decision function, or a nonlinear decision boundary, even when the data are linear non separable and this is realized by a concept

called the *kernel trick*. The kernel trick will be explained shortly but first the basic principle of a nonlinear SVM will be discussed.

In the left graph in Figure 8 a linear non separable data set is seen, which clearly is not linear separable and cannot be solved by a linear SVM, or at least not with good results. If some transformation function, related to as φ, is applied to all of the data points which transforms the data points into a higher dimensional space a linear separating hyperplane can be found. Note that the new dimensional space after the transformation can be of an infinitely high dimension. The process of applying a transformation function φ to all the data points is illustrated in Figure 8. In theory this means that we have the same problem as in the linear case which implies that a linear SVM could be used. Thus the process in order to find a linear separating hyperplane for a linear non separable dataset would be to first transform the data set into a higher dimensional space by some transformation function φ and then solve the linear problem. However, since the dimensional space might grow to infinity this process might not be computationally possible. This is instead solved by the kernel trick.

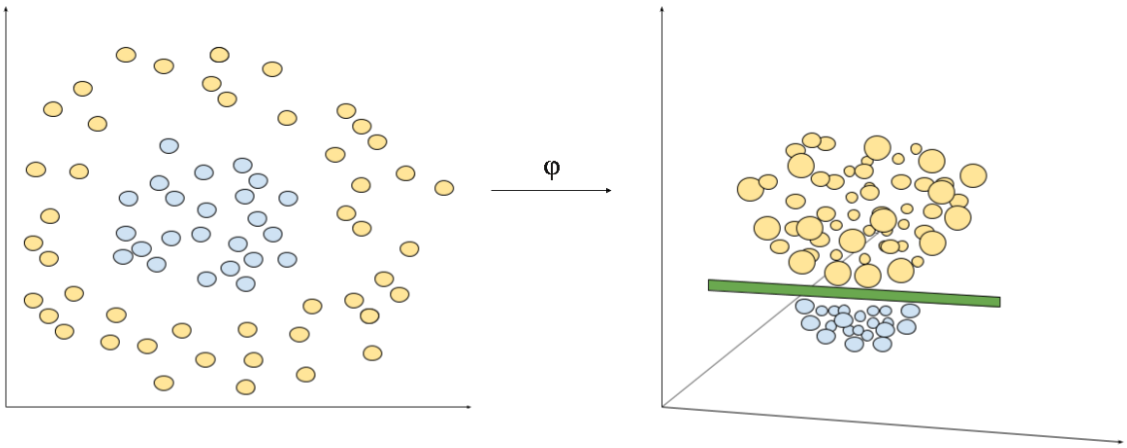

Figure 8 The left graph shows a linear non separable dataset. The right graph shows what the data set looks like in a higher dimensional space after some transformation function φ has been applied. In the higher dimensional space a linear separating hyperplane can be found.

In order to understand the kernel trick some knowledge about the underlying mathematics behind SVMs need to be known. However, a thorough explanation regarding the underlying mathematics of SVMs will not be covered in this report but suggested reading can be found in

[**26**] and [**27**]. In short the mathematics behind SVMs is about vector calculations and dot products and because of this it can be shown that during the training of a SVM model the problem of finding the most optimal hyperplane is computed by pairwise dot products. This is an important observation because there are functions that given a pair of vectors in $\mathbb{R}^N$ can implicitly compute their dot product in a higher dimensional space $\mathbb{R}^M$ without explicitly transforming the vectors to $\mathbb{R}^M$ [**28**]. These functions are called *kernel functions*. It is called the kernel trick because; (i) by using a kernel a dataset can be implicitly transformed into a higher dimensional space and (ii) with the help of a kernel function a nonlinear decision boundary can be learned by the classifier without, or with minimal, additional computational time.

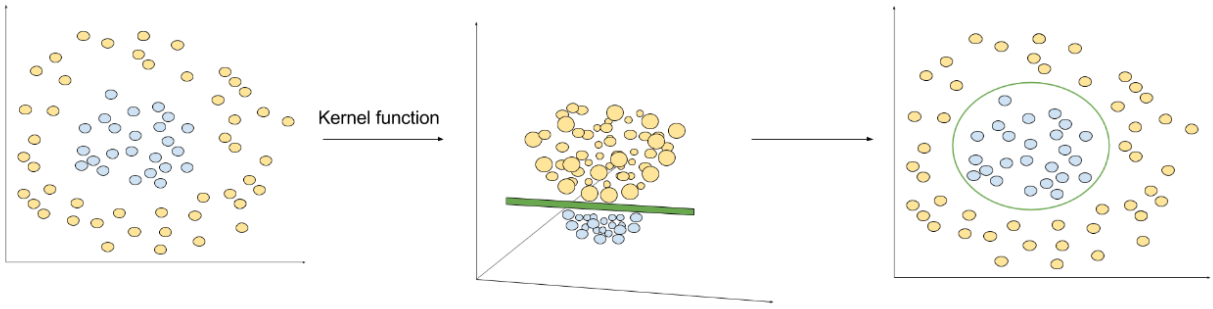

Figure 9 The process of the kernel trick in order to find a nonlinear decision boundary by an implicit transformation by a kernel function.

2.3.2. Decision Trees

Decision Trees are trees that reach conclusions about observations by sorting their feature values down a tree [**29**]. Each node in a decision tree, except for the leaf nodes, represents a feature in the feature space and the branches each represent a feature value that the node above (the parent node) can assume. The leaf nodes represent classes that the observations are classified to belong to. The main idea behind a decision tree is that an observation is classified by starting at the root node and is sorted based on its feature values down the tree and when a leaf node is reached the classification is determined.

As an example of a decision tree, assume there is a life insurance company that classifies people into two classes according to the estimated risk of dying an unnatural death. The two classes are "high risk" and "low risk". The insurance company wants to earn as much as possible, so they do not sell insurances to people classified as belonging to the class "high risk". In order to do the classification the two features *sex* and *age* are extracted, where the former feature can assume the values, "male" or "female", and the latter the values, "less than thirty" or "greater or equal to thirty", as can be seen in Figure 10.

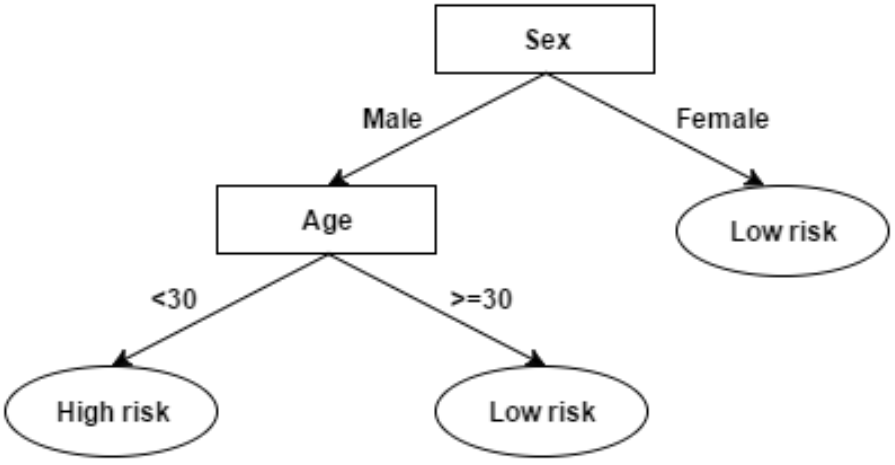

Figure 10 A Decision Tree for predicting a persons estimated risk of dying a premature death.

2.3.3. Ensemble Methods

Ensemble methods are a type of learning algorithms that instead of creating a single classifier composes a collection of individual classifiers [**30**], i.e. ensemble methods can also be thought of as combining a group of *weak learners* into a *strong learner* [**31**]. When predictions are made on new observations the individual classifiers predictions, i.e. the weak learners, are combined in some way, e.g. by weighted voting, in order to decide the final prediction. An ensemble of classifiers may give more accurate predictions compared to its individual classifiers and it is also inherently parallel which can reduce the time for training and testing phases if multiple processors are available.

An example of an ensemble method is Bootstrap Aggregation (Bagging) which uses bootstrap sampling to generate multiple training sets from the given data set [**32**]. Bootstrap sampling is a technique where samples are drawn randomly with replacement. All of these bootstrap samples, or training sets, are then used to train and obtain a number of classifiers. The predictions on new observations from all of these individual classifiers are then combined and by voting, i.e. in the case of classification, reach a final prediction.

Another example of an ensemble method is Random Forest which in principle works by averaging multiple Decision Trees, i.e. the Random Forest is the strong learner which is built from multiple weak leaners, the Decision Trees [**31**] [**33**]. Figure 11 show an example of a Random Forest that was built by three sub trees. Each sub tree was trained on roughly 2/3rds of the total training data and each sub tree uses a certain number of randomly drawn features, which for classification problems usually is the square root of the total number of features. The rest of the training data at each sub tree can used to calculate the misclassification rate. The misclassification rate can be used to assess the performance of the model. When prediction on new data is carried out each sub tree outputs a classification, i.e. each sub tree has one vote, and the forest of trees chooses the classification with the most votes.

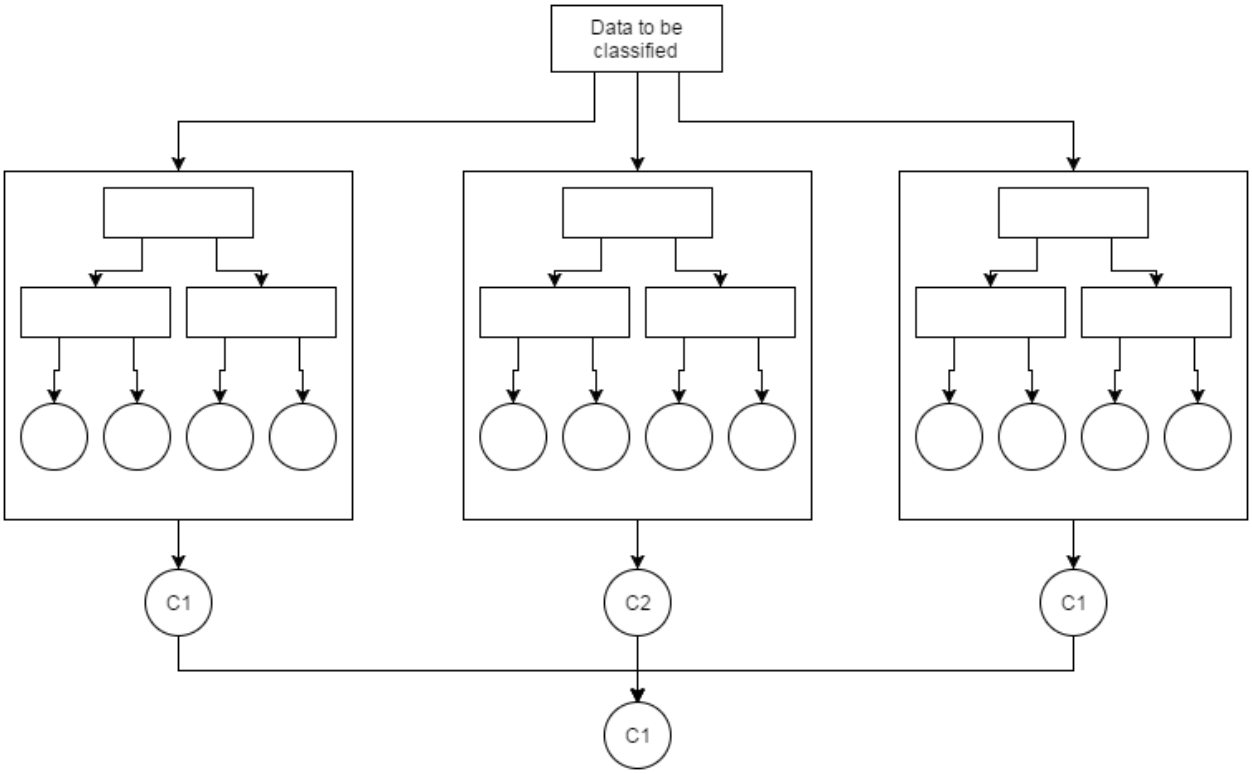

Figure 11 A Random Forest.

2.3.4. Instance Based Learning

Instance based learning algorithms are a type of *lazy learning* algorithms meaning that they do not generalize a model to the training data, i.e. instance based algorithms do not explicitly train a new classifier to carry out predictions on new observations. Instead instance based algorithms usually create a database with instances of training samples [**34**], which means that the "training" phase is minimal compared to other algorithms. However, this is also one if its greatest weaknesses due to the fact that it needs to store all of, or at least much of, the training samples in memory. When making predictions on new observations the model compares, or creates hypotheses, on new observations with the training samples using some kind of similarity measurement in order to find the best match and carry out predictions. This implies that the instance based learning algorithms complexity can grow with training samples, i.e. as the database grows the more complex the model becomes [**35**]. Assume that the database consists of *n* training samples; in the worst case scenario the hypotheses consists of *n* training samples and the computational complexity is *O(n)* when prediction is to be carried out on new observations, which may be slower compared to algorithms that have an explicit training phase. One of the

great advantages with an instance based model is its adaptability to new unseen data. Instead of having to train a new classifier in order to cope with new data as with other types of algorithms, such as SVM or Decision Trees, instance based models can simply store new training samples or throw away old ones.

An example of an instance-based learning algorithm is the k-Nearest Neighbor (kNN) algorithm [**36**]. A general explanation of the kNN algorithm is as follows: Find the *k* nearest neighbors i.e. the training samples, to the new observation and perform a majority vote between the neighbors deciding which class the new observation belongs to. In this very simple model all of the training samples contribute equally. However, this might not be optimal. There are extensions to the algorithm that for example takes the Euclidian distance between the training samples and the new observations into consideration.

2.3.5. Bayesian Methods

Bayesian methods are based on Bayes Theorem. Bayes theorem is a simple mathematical algorithm that is used to calculate conditional probabilities. A simple and easy to follow introduction can be found in [**37**] and further reading can be found in [**38**]. In short, the idea behind Bayes theorem is to give understanding about an entity given information about some known evidence. From a machine learning perspective this can be seen as to classify an observation given some known feature. The definition of Bayes theorem is given in Equation (1), where $P(A|B)$ and $P(B|A)$ are conditional probabilities and $P(A)$ and $P(B)$ are the probabilities of A respective B independently of each other.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{1}$$

One of the more simple Bayesian methods is the Naïve Bayes. If we look at Equation (1) from a machine learning perspective, A and B become class and feature respectively. However, when considering more than one feature, it gets a lot more complicated. In order to overcome this complication, an assumption can be made, namely; the features used are independent of each other, i.e. no correlation between the features exists, hence the name Naïve Bayes. With this assumption it boils down to Equation (2), where C_k are all the possible classes and $\boldsymbol{f}$ is a feature vector, i.e. $\boldsymbol{f} = (f_1, f_2, \dots, f_n)$.

$$P(C_k|\boldsymbol{f}) = \frac{P(f_1|C_k) * P(f_2|C_k) \dots P(f_n|C_k) * P(C_k)}{P(f_1|C_k) * P(f_2|C_k) \dots P(f_n|C_k)} = \frac{P(\boldsymbol{f}|C_k) * P(C_k)}{P(\boldsymbol{f})} \quad (2)$$

The Naïve Bayes method is applied by running the above equation, Equation (2), for every possible class. The goal is to classify new observations by looking at how likely it is that it belongs to a specific class.

2.4. Chapter Summary

In this chapter machine learning was introduced followed by a detailed discussion about the SVM. The SVM was the method used during the experiments in this study which results are presented in Chapter 6 and in Appendix A. Several other machine learning methods were discussed, including Decision Trees and Random Forest, which were the methods used in the parallel study.

3. Related Work

Throughout the recent years research regarding early Internet traffic classification has progressed and received more attention due to the fact that more and more network applications tend to encrypt and/or use other obfuscation techniques. There have been several studies regarding Internet traffic classification which show promising results and some of them are briefly discussed in this chapter.

The work of Lizhi Peng et al. [**8**] had the purpose to examine the effectiveness of statistical features when early Internet traffic classification is to be carried out using different machine learning methods. Statistical features are features that are computed based on a certain number of packets. Ten different machine learning methods were used and among them were the k-Nearest-Neighbors (k-NN), Bagging and random forest algorithms. In their study six different feature sets were used based on the first six non-zero payload packets. The feature sets consisted of a combination of the packets payload sizes and five statistical features; average payload size, standard deviation, maximum and minimum payload sizes, and variance. Three different data sets were used and among them were traffic classes such as web browser, stream media, mail and P2P. According to their findings the majority of the machine learning methods achieved an accuracy of around 95% for all the three data sets that were used and they also concluded that most of the statistical features, except for the minimum packet payload size, were as effective as the packet payload sizes.

Jeankyung Kim et al. [**9**] present a novel classification method that uses Kullback-Leibler information and in their study they evaluate two models based on their novel classification method; KL-10 (Kullback-Leibler with bag size 10) and KL-100 (Kullback-Leibler with bag size 100). They utilize a technique where they group together traffic flows that are believed to belong to the same application into a bag of flows and then they classifies the whole bag instead of individual flows, hence the name of the models. The main purpose of their study was to classify hard-to-classify Internet traffic by only using the first few packets. They use only the first four packets, excluding the TCP handshake and acknowledgments, in their classification, note that their study only considered TCP traffic. They first tested and evaluated using IMAP and SMTP traffic, which are very similar to each other considering their traffic patterns, therefore considered hard-to-classify, and their results shows that both models reaches a recall close to or

above 90%. After their evaluation using only IMAP and SMTP traffic, they extended their models in order to be able to classify other traffic classes as well. Their extended models were applied to traffic generated from a total of ten different network protocols, among those were FTP, HTTP, POP and HTTPS. After the extension they show very promising results again for their novel classification method.

Talieh Seyed Tabatabaei et al. [**3**] present a work in which two supervised machine learning methods, SVM and k-NN, are evaluated and compared. They used two different implementations of SVM, fuzzy one-against-all and fuzzy pairwise. In their study they looked at seven different types of Internet traffic, which consisted of e.g. Bit Torrent, web browsing and Skype traffic, and tried to classify the traffic using only the first few packets, varying the number of packets between three and fifteen. They utilized a program called NetMate in order to generate traffic flows and to extract and compute their features. A total of forty features were used in the initial feature set and the features were computed separately for each direction, i.e. the two directions from host to client and from client to host. The main features were inter-arrival times, number of packets, flow duration, packet lengths and number of bytes. They used a technique called minimum redundancy-maximum relevance (MRMR) in order to choose which subsets of features to feed the machine learning algorithms in the training phase. Their result shows that the fuzzy one-against-all SVM method when considering the first seven packets was the most accurate one, achieving an accuracy of 84.9%. However, the k-NN method was not far behind with an accuracy of 84.28% while the fuzzy pairwise SVM method was the one that performed worst, achieving an accuracy of only 74.65%. Note that all of the methods achieved a better accuracy when considering the first seven packets compared to the entire flows.

Chengjie GU et al. [**39**] present a work which purpose was to classify Internet traffic within the first few packets using flow statistics. They proposed a classification method based on proximal SVM and compared their method with three other SVM based methods. In short, the proximal SVM method is a simplification of the standard SVM method, which generally makes it considerably faster. In their study, accuracy is defined as the sum of true positives divided by the sum of true positives plus the sum of false positives. In their evaluation and testing, three different datasets were used with ten-fold cross validation applied. The largest of the three data sets consisted of a total of 7.15 million traffic flows and some of the traffic classes were P2P, media, game and mail. For the largest of the three data sets, using only the first ten packets, the

proximal SVM method achieved an accuracy of 90.58%, which were the lowest accuracy of all four methods, compared to the SVM which achieved the highest accuracy of 91.62%. However, the computational time was considerable faster for the proximal SVM method which needed 75.3 seconds, compared to the SVM with the highest accuracy which needed 812.7 seconds. In their study, they examined how much the number of packets affected the achieved accuracy, and came to a conclusion that ten packets were their golden number; under ten packets, the accuracy got worse, while over ten packets the accuracy only improved slightly but since a classification was to be carried out as fast as possible, and with good performance, a decision to use ten packets was made. Unfortunately, no documentation about which features were used could be found.

Alberto Dainotti et al. [**4**] performed a study in which they used six different combination algorithms, i.e. algorithms that combine several machine learning classifiers, with eight different classifiers. A total of six different machine learning classifiers were used in addition to one port based and one payload based, in order to explore how well a combination of classifiers could perform when early Internet traffic classification is of interest. They first present the standalone accuracies for each and every classifier based on a data set consisting of twelve different applications. Some of the applications were bit torrent, skype, HTTP and eDonkey. In their study they used features such as payload sizes, packets inter arrival times and statistical features derived from these two features and also flow durations. The results show that four of the machine learning approaches, Decision Tree, kNN, Random Tree and Ripper, achieved accuracies between 95.9% and 97.2% while the Naïve Bayes only reached an accuracy of 43.7% and the port based, even worse, only achieved an accuracy of 15.9%. They also show their assessed theoretical accuracy of 98.8% for the combination of these standalone classifiers. The result by combinations of different classifiers achieves even higher accuracies than the individual ones. They also show results for accuracies achieved when considering number of packets in the range from 1-10 for both the standalone classifiers and the combinations of them. According to these results when considering packets in the range from 1-3 the accuracies can greatly be improved by a combination of classifiers.

Jayeeta Datta et al. [**10**] performed a study in which they used three different machine learning algorithms in order to try and classify encrypted traffic. The algorithms evaluated were Naïve Base, AdaBoost and J48 decision tree. In their study they divided their collected data into four groups; Google Hangout, Google Plus, Gmail and Not Hangout. Google Hangout is an

application that allows its users to do text chatting, voice over IP and video chat in real-time through a conference server. Several hours' worth of Internet traffic was collected and it comprised a total of 2.5 million packets. Instead of classifying the traffic flows based on flow characteristics their extracted features were per packet based. A total of seven features were extracted from the data and in order to understand these features the network protocol Session Traversal Utilities for NAT (STUN), which is used by Google Hangout, need to be introduced. STUN is a network protocol that allows end hosts to discover their public IP address if they are located behind a NAT. Some of the features were "Layer 4 protocol", "Is UDP using same Source Port as STUN", "Destination Port Number", "Packet Length" and "Type of STUN Packet". A total of three different test cases were conducted in which the following groups were classified; (i) Hangout and Not Hangout, (ii) Hangout, Not Hangout and Gmail and (iii) Hangout, Not Hangout, Gmail and Google Plus. In the first test case both J48 and AdaBoost reached a recall of 99.99% while Naïve Base reached a recall of 97.91%. In the second test case AdBoost reached a recall of 99.99% closely followed by Naïve Base on 99.98 % while J48 reached 99.46%. In the last test case both AdaBoost and J48 reached a recall of 100% and Naïve Base 99.98%.

In the study by Li Jun et al. [**40**] five different machine learning algorithms were evaluated. The performances of the classifiers with regard to the dataset size and feature selection were demonstrated in tests. The five algorithms evaluated were Tree augmented Naïve Bayes (TAN), C4.5 Decision Tree, Naïve Bayes Tree, RandomForest and kNN. The feature selection was done by a generic algorithm (GA) which after a series of iterative computations reaches the optimal selection. They simulated traffic with around 100 hosts in order to generate the test data and then labeled the traffic flows, calculated the flows features and lastly sampled the test data so that each application were equally represented by number of flows. Some of the applications considered were HTTP, FTP, streaming, game, skype and DNS. Several features were computed in the initial feature set and some of those were flow duration, total number of packets of the flow, inter arrival time of the packets, packet payloads. After the feature selection by the GA the subset of features was flow duration, total bytes of flow, the maximum packet inter arrival time and minimum packet payload size. In their first test using a data set consisting of 10400 samples in total, they show that the overall accuracy when using the subset of features compared to all of the features in the initial feature set were slightly reduced. However, the time during training and

prediction were significantly reduced. The Naïve Bayes tree went from an accuracy of 95.28% to 94.86% while the training time was reduced from 1922 seconds to 305 seconds and the prediction time was reduced to 29 seconds from 158. In their second test they show that as more data are used during the training phase the classification performance tend to improve but according to their results it seems to steadily settle down as the test data grows large. In their last test their results show that as the number of Internet traffic classes increases the overall performance of the classifiers decreases.

Jaiswal Rupesh Chandrakand and Lokhande Shashikant D. [**41**] performed a study in 2013 in which six different machine learning algorithms were compared; AdaBoostM1, C4.5 Decision Tree, Random Forest, Multilayer Perceptron (MLP), Radial Basis Function Neural Network (RBF) and SVM. The data set used was real time internet traffic captured by a tool named Wire Shark which consisted of nine different categories of Internet traffic. Among the nine different categories were DNS, gaming, skype, streaming and torrent. In their experiments two data sets were used; the first data set consisted of all the captured data and the second data set was a sampled data set derived from the captured data. In their study a total of 10 statistical features were computed and these were packets/sec, mean packet length, mean payload length, total bytes (flow), flow duration, mean inter arrival time, bytes/sec, standard deviation for packet lengths, inter arrival time and payload lengths. According to their study the Random Forest classifier reached an overall accuracy of 99.76% for both data sets closely followed by the C4.5 classifier with an accuracy of 98.47% and 98.46% for the whole and sampled data set respectively. The Random Forest and C4.5 classifiers were also the two classifiers including AdaBoostM1 that needed the least amount of training time. However, AdaBoostM1 was the classifier with the worst accuracy. The RBF classifier was by far the most time consuming classifier and it only reached an accuracy of about 84% for the sampled data set.

Zhu Li et al. [**42**] conducted a study where the SVM with RBF kernel were used to classify 7 different classes of applications among which were WWW, P2P, mail and bulk. All of their test data were captured within one week and comprised 8 hours' worth of traffic data. Two different feature selections methods were used: The first one was sequential forward selection in which 0 features are used to begin with and then 1 feature is sequentially appended until the best combination is reached. And the second feature selection method was an extension of the first one. A total of 19 different features were computed where average packet size of the flow,

number of packets sent in the flow, the duration of the flow and the variance of the size for packets sent were some of the features. According to their result the influence of the value of the penalty parameter C regarding the SVM method was negligible and they used a value of 2000 for C. In their first test when all of the 19 features were used an overall accuracy of 99.4146% was achieved and after feature selection by the extended sequential forward method, which reduced the number of features to 9, an overall accuracy of 99.4167% was achieved. As seen the overall accuracies were about equal, however, the false negative values were decreased after the feature selection.

In a study made by Kuldeep Singh and Sunil Agrawal [**43**] five machine learning algorithms were evaluated regarding Internet traffic classification using a real time captured data set. The algorithms they used were MLP, RBF, C4.5 Decision Tree, Bayes Net and Naïve Bayes. They used the tool Wire Shark to capture their data and for each application considered 2 minutes of data were captured. The applications considered were P2P, WWW, VoIP, web media etc. A total of two datasets were derived and in both data sets the total number of samples was 2800. The first data set was described by a total of 261 features and in the second data set a reduced number of features were used. The features were based on the flow as measured in time rather than based on a specific number of packets. Some of the features were; statistical features derived from number of packets, e.g. mean and variance, average packets per second, packet sizes and duration of flow etc. During their experiments they used 2500 samples for training and 300 samples for evaluation for both of their test sets. According to their results the Bayes Net classifier was the classifier that reached the highest overall accuracy when the full feature data set was considered with an accuracy of 85.33%. Regarding the reduced feature data set the C4.5 Decision Tree classifier reached an accuracy of 93.66%, which was the highest score, while the c4.5 Decision Tree only reached an accuracy of 79% considering the full feature data set. They also show recall and precision values regarding the different applications for the three classifiers that performed the best, namely the RBF, C4.5 Decision Tree and Bayes net classifiers. The C4.5 classifier reaches a recall and precision of 100% for 7 out of the 9 applications followed by the Bayes Net classifier with 7 and 6 applications respectively.

3.1. Chapter summary

In this chapter several similar studies has been discussed. Various machine learning methods were used and the classifiers performances between the studies differ a lot. This is most likely the result of different data sets with different traffic classes being used where some of those data sets may be easier to classify. Furthermore, in each study different features were computed. Note that in some of the studies the features were computed based on entire flows instead of computed for a certain number of packets. Features based on entire flows could not be used in this study since a decision was made to use packet based features. However, the flow based studies are still interesting since it is another way of approaching the traffic classification problem.

4. Design of Study

In this chapter of the report the sampled data sets that were used during the training and evaluation of the classifiers will be presented followed by various evaluation metrics that was used to assess the performances of the classifiers. A technique called *cross-validation* will be introduced. And lastly the two different implementations of the feature sets will be discussed.

4.1. Data Sets

The first data set that Procera provided was a small data set that were mainly used in order to get an overview of what the data looked like and to figure out how the preprocessing of the data was to be carried out. In the first data set there were not a great deal of video traffic to use during the training and testing of the classifiers and because of this there are no results presented regarding this data set.

The second data set that Procera provided consisted of 37 smaller files that together constituted a large data set that were captured in one of Proceras boxes over a time interval of around an hour. In order to be able to work with the large data set in an efficient and easy way there was a need to reduce the size considerably; all of the raw data files added up to a total of ~200 gigabyte of data. The data set was reduced by removing some data that were unnecessary during this study and by changing data types for the various entries.

From the larger of the data sets a total of five sampled data sets was produced in different sizes where two of these data set were used during this study. The sampling was done in such a way that the sampled data sets were represented by a certain amount of video traffic in addition to other kinds of traffic flows. In the following figures summarizing the data sets the top ten most represented Internet services are shown. The names of the different sampled data sets are denoted as following: *Vk_Ok*, where *V* stands for the total number of unique video flows and *O* the total number of other types of traffic flows.

4.1.1. 10k_10k

```
STATS for Unfiltered
Total Number of Packets  :  1078706
Number of Unique Flows   :  20707
Number of Unique Services:  134

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |      7617 | 36.78
SSL v3                                  |      2768 | 13.37
HTTP                                    |      2260 | 10.91
BitTorrent transfer                     |       880 | 4.25
uTP                                     |       769 | 3.71
Youku                                   |       700 | 3.38
MiBox                                   |       678 | 3.27
Being analyzed                          |       616 | 2.97
Unknown                                 |       498 | 2.40
DNS                                     |       422 | 2.04

Freq Video

SERVICE   |Count      | Share (%)
_____________________________
Video     |     10359 | 50.03
Other     |     10348 | 49.97
```

Figure 12 The sampled data set 10k_10k.

4.1.2. 100k_100k

```
Total Number of Packets  :  10589858
Number of Unique Flows   :  203940
Number of Unique Services:  232

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |     75245 | 36.90
SSL v3                                  |     26427 | 12.96
HTTP                                    |     21999 | 10.79
BitTorrent transfer                     |      8956 | 4.39
uTP                                     |      7568 | 3.71
Youku                                   |      7169 | 3.52
MiBox                                   |      6357 | 3.12
Being analyzed                          |      6029 | 2.96
Unknown                                 |      4694 | 2.30
DNS                                     |      4249 | 2.08

Freq Video

SERVICE   |Count      | Share (%)
_____________________________
Video     |    102567 | 50.29
Other     |    101373 | 49.71
```

Figure 13 The sampled data set 100k_100k

4.2. Evaluation Metrics

It is very important to know how good a classifier performs in order to know which classifier too chose. Classifiers can be compared by their predictive accuracy, i.e. how accurately a classifier can predict when presented data that it has never seen before. There are some different metrics with which classifier accuracy can be expressed and some of these metrics, i.e. metrics that have been used in this study, will be covered in this section. One of the more common metrics used to express classifiers *accuracy* is through the metrics *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) and *False Negative* (FN). Three other metrics that are used is *Precision, Recall* and *Specificity* and lastly there is the Receiver Operator Characteristics (ROC) curve.

4.2.1. TP, TN, FP, FN and Accuracy

In order to define and understand TP, TN, FP and FN, assume there is a classifier that either classifies traffic flows to belong to class x, which in this case is considered to be the correct class, or classifies traffic flows not to belong to class x. Figure 14 illustrates the relationship between TP, TN, FP and FN in a confusion matrix. A confusion matrix is a useful tool that can be used to illustrate the outcome of a model in an intuitively manner and it can be used to derive many other useful evaluation metrics.

		Predicted class	
		X	$\overline{X}$
Actual class	X	True Positive	False Negative
	$\overline{X}$	False Positive	True Negative

Figure 14 The relationship between true positive, true negative, false positive and false negative illustrated in a confusion matrix. The green and red color represents good and bad outcome respectively.

Another means of describing TP, TN, FP and FN is to consider not just single flows of traffic, as in Figure 14, but rather describe them looking at the whole collection of flows.

- *True Positives* are defined as percentage of traffic flows correctly classified as belonging to class x.
- *True Negatives* are defined as percentage of traffic flows correctly classified as not belonging to class x.
- *False Positives* are defined as percentage of traffic flows incorrectly classified as belonging to class x.
- *False Negatives* are defined as percentage of traffic flows belonging to class x incorrectly classified as not belonging to class x.

Generally speaking, accuracy is defined as correctly classified entities among the total number of entities [**2**]. Equation (3) and Equation (4) defines accuracy through the metrics, TP, TN, FP and FN, discussed above.

$$accuracy = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN} \tag{3}$$

$$accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} \tag{4}$$

4.2.2. Precision, Recall and Specificity

If the same classifier as in section 4.2.1 is considered, precision, recall and specificity can be defined through the metrics TP, TN, FP and FN as in Equations (5), (6) and (7) respectively.

- Precision is defined as percentage off correctly classified traffic flows belonging to class x among the total number of traffic flows classified as belonging to class x.

$$precision = \frac{\sum TP}{\sum TP + \sum FP} \tag{5}$$

- Recall is defined as percentage of traffic flows correctly classified as belonging to class x among the total number of traffic flows belonging to class x.

$$recall = \frac{\sum TP}{\sum TP + \sum FN} \tag{6}$$

- Specificity is defined as percentage of traffic flows correctly classified as not belonging to class x among the total number of traffic flows not belonging to class x.

$$specificity = \frac{\sum TN}{\sum TN + \sum FP} \tag{7}$$

4.2.3. Receiver Operating Characteristics

A Receiver Operating Characteristic (ROC) space is a graphical plot where the x- and y-axes depicts the true positive rate (TPR), which is the same as recall, and false positive rate (FPR), which is the same as $1 - specificity$, respectively [**44**]. A point in the ROC space is drawn from a single models prediction result.

A ROC curve is a plot in the ROC space that can be used to illustrate the performance of a binary classifier as can be seen in Figure 15. In addition to the ROC curve the area under the curve (AUROC) is another metric that is useful, which is a measure of discriminative power. Consider a scenario where all observations are already classified correctly into two different classes, then the AUROC is the expectation that if two randomly drawn samples from each class are drawn and predicted again are classified correctly [**45**]. Note that a single model has a ROC curve, i.e. the model can produce a ROC curve while some threshold parameter is varied. You can say that there are two types of classifiers; either a classifier predicts class membership, i.e. a discrete classifier, or a classifier gives an estimate of how much an observation belongs to a specific class, i.e. a probabilistic classifier.

A discrete classifier produces one point in ROC space, which implies that a discrete classifier cannot produce a ROC curve by default. However, there are techniques that can be used in order to enable a discrete classifier to output class probabilities rather than class membership. In scikit-learn regarding some classifiers that by default only predicts class membership, like the Support Vector Machine, Platt scaling is used in order to enable class probability estimates in the case of

binary classification [**46**]. A short introduction to Platt scaling may be found in [**47**] while a more extensive coverage may be found in [**48**]. And regarding the multi-class classification problem a further extension is needed in order to plot a ROC curve which is explained in [**49**].

A probabilistic classifier can produce more points in ROC space by thresholding it to become a discrete classifier. Assume that a probabilistic classifier outputs an estimated probability X, which is a score for an observation that says how much the observation belongs to a specific class. By defining and using a threshold parameter T the probabilistic classifier produces a discrete classifier by saying that an observation is either correctly classified if $X > T$ or otherwise incorrectly classified which implies that each threshold value produces a single point in ROC space, i.e. the same as a discrete classifier.

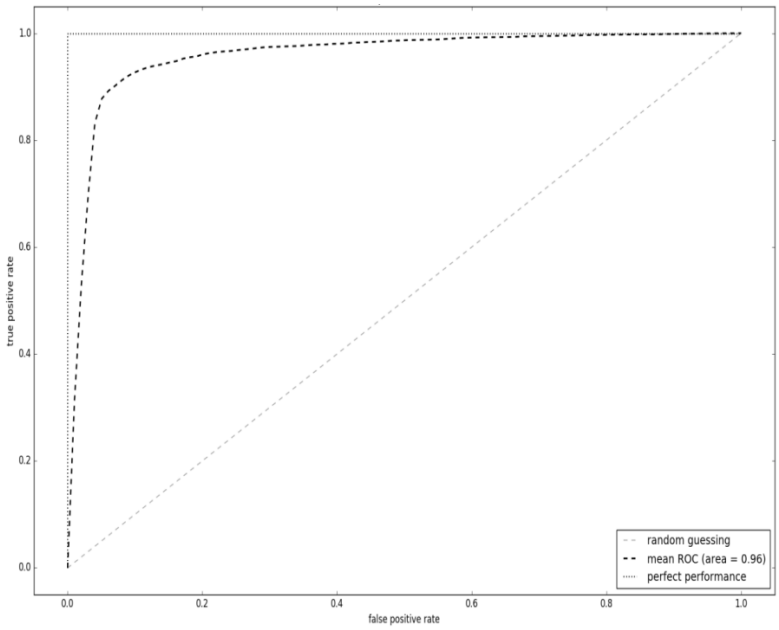

Figure 15 The ROC curve for a binary classifier.

4.3. Cross-Validation

When evaluating machine learning models, training and testing should be done on different data in order to achieve credible results. The data is typically divided into training and testing data, but at the same time, it is desirable to use all of the data for both training and testing. However, that is not possible. There are good techniques that can be used with which a good approximation of using all of the data can be achieved, and one of them is *cross-validation* [**50**]. The basic form of cross-validation is called *k-fold cross-validation,* which is the form of cross-validation that has been used in this study.

When using k-fold cross-validation, the data set is divided into k data segments, these segments are also known as *folds.* Each fold is divided into two segments, one for training and the other for validation, as can be seen in Figure 16. After the data set has been divided into k folds, k iterations of training and testing are done. The performance of a specific model is typically expressed as the mean of the achieved performances of the models during the k iterations. The idea with cross-validation is that each data point is used for both training and validation, but not in the same iteration, i.e. not in the same fold.

Figure 16 5-fold cross-validation.

4.3.1. A Common Mistake when using Cross-Validation

A common mistake that is important to be aware of when using cross-validation is best understood by an example. Assume that a data set with 100 samples is at hand and each sample is described by 500 features. Then before any training and evaluation of a model some feature selection step is typically executed to reduce the number of features. Essentially to evaluate a model the process can be described by 2 steps; (i) selection of the *X* best features and (ii) training

and evaluation of the model with the *X* selected features. A common mistake is to in step (i) select the *X* best features in regard to the data set as a whole. And apply cross-validation in step (ii) to train and evaluate a model. But in this case in the selection step of the *X* best features all of the class labels have been exposed and the features chosen are too good. Instead the cross-validation should have been applied to both step (i) and (ii) so that the features chosen only are based on part of the data. If more information about the topic is desired the video found in [**51**] is suggested, which the above discussion was based on.

4.4. Preprocessing

The machine learning methods used through the scikit-learn modules takes as input a numpy array of a certain shape that can only contain features of numerical values. However, the class labels in the training phase can be either strings or numerical values. Because the machine learning algorithms only take numerical values as input, features such as categorical features need to be encoded somehow. Categorical features are features that have no apparent numerical representation, the color of an object or the brand of a car for example. In this study the direction of the packets was treated as a categorical feature.

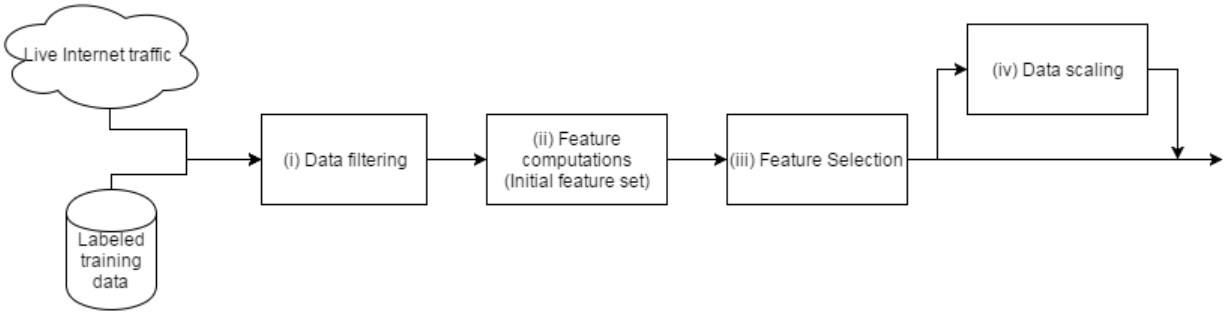

Figure 17 The preprocessing process of a data set.

The preprocessing of the raw data can be divided into four parts, as can be seen in Figure 17: (i) filtering of the data set, i.e. remove flows and/or packets that are not to be used during the training and validation of the model. This includes flows that lack ground truths, flows that are caught mid-stream, i.e. flows caught after the recording of the Internet traffic started, and other defined constraints that can be tuned. For example, if a certain number of packets are to be considered then the rest can be removed, or if packet sizes of a specific size are deemed uninteresting those can be removed. (ii) The computation of features used in the initial feature

set, i.e. features that describe the raw data in the best possible way. (iii) The selection of the best features and lastly (iv) scaling of the data, which is sometimes omitted depending on which machine learning method is used. Note that when evaluating different models, all four, or at least the first three, of the above steps are covered. That is, when a machine learning method is chosen and implemented into a live Internet traffic classification system, only the two first steps and sometimes also the fourth step need to be considered. Actually, since at that point the best features are known, only a modified version of step two needs to be executed, i.e. only the chosen features needs to be computed. However, this section will briefly cover how the preprocessing of the raw data was done in the evaluation phase in order to achieve a representation of the data that the machine learning methods could use.

In this study two different initial feature sets were derived; the first one containing features which only consider the payload packets of the traffic flows and the second one containing features considering all of the packets in the traffic flows. The two different feature sets are from this time on called *the payload feature set* and *the packet feature set* respectively. However, since the two feature sets consider different kinds of traffic this implies that the above discussion of the preprocessing phase was executed differently for each of the two feature sets. Each of the following subsection will thus be divided into two parts, one for each of the feature sets.

4.4.1. Data filtering

4.4.1.1 Payload Feature Set

In this approach, as the name suggests, only payload packets were to be considered during the evaluation; mainly because it was seen in similar studies to be effective but also because of the nature of TCP acknowledgements for example which will be discussed shortly. In order for the features to be consistent throughout the evaluation of a model, i.e. all the flows are to be considered equal, some flows had to be removed, e.g. traffic flows that did not have at least the desired number of payload packets.

Since early Internet traffic classification is of interest, the first numbers of packets in the transmissions are to be considered, i.e. the first payload packets. This implies that the first step in the filtering process was to remove flows that are caught in the middle of their transmissions. In order to be able to evaluate a model, the service for each traffic flow needs to be known, which

implies that traffic flows that lacked ground truth had to be removed. Here the term “ground truth” means that a traffic flow is known to be labeled with the correct service.

TCP acknowledgements can in general be seen as random in nature, i.e. there is no certainty at which point in time in a transmission an acknowledgement is to be received. This means that TCP acknowledgements may not contribute during a classification. Thus a decision was made that only packets with payload sizes are to be considered, this has also been seen to be used in similar studies, see Chapter 3. However, since the actual payload of the packets cannot be seen, there is no way to only remove packets that do not carry any payload, e.g. TCP acknowledgements. Instead it was decided that it would suffice to remove packets less than a certain size. In this study packets above 70 bytes were considered payload packets.

As earlier mentioned, all the traffic flows were to be considered equal, which means that traffic flows that did not reach a certain number of payload packets was not to be considered so these flows were simply removed.

Since several similar studies have used payload sizes for the first few packets as features, up to the first 15 payload packets, with good results, the payload sizes are features that should not be ignored. Thus there was also a need to be able to filter packets in flows when the desired number of packets was reached leaving only the first number of packets that was to be considered during the evaluation.

As mentioned earlier, one of the goals of this study was to explore if machine learning algorithms are able to classify encrypted traffic. However, in order to evaluate machine learning models the traffic flows need to have ground truths. This implies that encrypted traffic flows, which are not explicitly given ground truths, cannot be used during the evaluation of a model.

4.4.1.2 Packet Feature Set

Compared to the other approach only considering the payload packets, this approach takes all of the packets in the traffic flows into consideration. The filtering regarding this approach was much simpler and the functionality already developed could be reused.

The same as for the payload approach regarding the ground truths of traffic flows and encrypted traffic applies to this approach as well. A decision was made to remove traffic flows

that had less than ten packets, because they were considered to be too short in order to be video traffic.

4.4.2. Initial Feature Sets

In this sub-section all of the different features that were considered during the study will be presented. There could be potentially infinitely many features but recall that this study was not about finding the best features, rather to explore the ability to classify Internet traffic. With that in mind, just a couple of hand-picked features were derived which were deemed to have discriminative power. All of the features presented below are computed relative to a single unique flow.

4.4.2.1 Payload Feature Set

Features used in the payload feature set are features that have been suggested by related works to be effective.

- Payload sizes, abbreviated *ps,* for each of the individual packets considered. Recall that only packets containing a payload were considered.
- The inter arrival time, i.e. the time elapsed between each payload packet. This feature was calculated separately for each direction.
- The direction of the packets. As mentioned earlier, this was seen as a categorical feature and thus it needed to be encoded somehow. A decision was made to use *one hot encoding* which means that if *x* packets are to be considered then the direction of the packets are described by 2^x new features. This is best understood with an example. Assume that 2 packets are to be considered which means that there are four possibilities for the sequence of the packets to traverse a network. For example as seen from the client side the four possibilities are: (i) up-up, (ii) up-down, (iii) down-up and (iv) down-down. Where "up" denotes that the client sends a packet and "down" that the client receives a packet. Assume that a flow has the packet sequence up-down then the direction of the packets is coded as in Table 1.

Table 1 One hot encoding.

Feature name	Up-up	Up-down	Down-up	Down-down
Feature value	0	1	0	0

- The average payload size. The definition is seen in Equation (8), where *i* is the packet index and *n* the total number of packets.

$$avg\_ps = \frac{\sum_i^n ps_i}{n} \tag{8}$$

- The variance of the payload sizes. The variance measures how much the payload sizes are spread out, a value of zero means that all of the payload sizes are equal. The definition of variance can be seen in Equation (9).

$$var\_ps = \frac{1}{n-1}\sum_i^n (ps_i - avg\_ps)^2 \tag{9}$$

- The standard deviation of the payload sizes. The standard deviation is the square root of the variance and it measures how much the payload sizes deviates from the mean. The definition is given in Equation (10).

$$std\_dev\_ps = \sqrt{\frac{1}{n-1}\sum_i^n (ps_i - avg\_ps)^2} \tag{10}$$

4.4.2.2 Packet Feature Set

The packet feature set contains all of the features which the payload feature set contains, except for the features describing the direction of the packets. And note that these features were also computed based on a certain number of payload packets only, the same as in the payload feature set. The packet feature set also includes the additional features presented below which were derived with the help of the supervisor. These features were decided by inspection of the data that they may be effective according to how the characteristics of the different categories of Internet traffic flows looked like. All of the features described below are either computed for the downward direction of the packets in the flow, i.e. packets arriving at the client side of an

application for example, or in the upward direction, i.e. packets sent from the client for example. Furthermore, the features were calculated at specific packet indices within a specified interval and a step size of two. For example, assume that the specified interval is 6 to 10 then the features were calculated for packet indices 6, 8 and 10 respectively, resulting in three separate features. The following is the list of features that were considered, in addition to the payload features discussed above.

- Number of packets in the downward direction.
- Number of payload packets in the downward direction.
- Number of bytes in the upward direction.
- Number of bytes in the downward direction.
- Number of payload bytes in the downward direction.
- The position sum of packets in the downward direction. The position sum is calculated by summarizing the packet indices.
- The position sum of packets in the downward direction with even indices.
- The position sum of payload packets in the downward direction.
- The position sum of payload packets in the downward direction with even indices.
- The longest consecutive run of downward payload packets.
- The longest consecutive run of downward payload packets where the inter arrival time of the consecutive packets are less than a certain amount of time, e.g. 200 ms.

4.4.3. Scaling The Data

Scaling the data means changing the unit of measurement [**52**]. An example of scaling could be to convert pounds into kilograms. Scaling is typically done by adding or subtracting a constant to a value and then multiplying or dividing the value by a constant.

When working with machine learning algorithms in the scikit-learn module scaling, or standardization, of the datasets may be a requirement in order for the algorithm to behave as expected [**53**] [**54**]. When working with SVMs it is very important to scale the data before

feeding the data to the SVM algorithm due to the fact that SVMs assume that the data is in a standard range. That the data is in a standard range means that each feature value is reshaped, or scaled, to lie within a specific range, e.g. in the range 0 to 1, or such that the individual features has zero mean and unit variance. For example the SVM algorithm with rbf kernel assumes that all features in the feature vectors are centered around zero and that the variance is of the same order. Let us consider two features in a data set where in the first feature the values ranges from 0 to 1 and in second feature the values ranges from 0 to 1000, then the second feature might have a bigger impact in the training and prediction in the classifier compared to the first feature, which may not be desirable and might lead to skewed results.

There are two clear advantages of scaling the data: (i) As mentioned above, features which values are in a greater magnitude than other features get toned down so that their impact on the classifier will be close to equal to all the other features. (ii) The second advantage is that numerical difficulties can be avoided, i.e. very large values may result in numerical problems when computations are carried out. For example, some kernel functions depend on the inner product of feature vectors which for large values might cause numerical problems.

Another important aspect to take into consideration regarding scaling data when evaluating a model is at what point in the process to perform the scaling transformation. There are essentially two possibilities; either the scaling of the data is considered a preprocessing step, which means that the scaling is done on the data set as a whole, or the scaling can be done on the training and testing set separately and is then considered to be a part of the learning procedure. However, when using a scaler, as the latter approach suggests, it is important to apply the same scaling transformation on the testing set as was performed on the training set.

4.5. Chapter summary

In this chapter the evaluation metrics accuracy, recall, precision and the ROC curve have been defined followed by a short introduction to the concept of cross-validation. The preprocessing of the raw data set including the filtering of the data set, the two implementations of the initial feature sets, i.e. the payload features set and the packet feature set, have been presented. In addition the scaling process of data has been discussed.

5. Scikit-learn tools and algorithms

In this chapter the feature selection algorithms that have been in consideration during the study will be briefly discussed. In addition the two machine learning algorithms, Linear SVC and SVC, which were used in this study will be presented. Note that SVC is the same as SVM (in Scikit-learn the SVM algorithms are simply called SVC).

5.1. Feature selection algorithms

In this section the feature selection algorithms that have been used to some extent will be discussed briefly. All of the feature selection algorithms that have been used are implemented in the scikit-learn module.

5.1.1. Selection by variance

Feature selection by variance involves removing features where the feature values between samples vary within a specified threshold. The default is to remove features where all of the samples have the same values [**55**].

5.1.2. Univariate selection

Univariate feature selection is a very simple form of feature selection in which at most one variable is considered at any time [**55**]. This implies that it may not be very powerful at distinguishing which features are the best to use since no combination and/or relation of the different features are considered by using this kind of selection algorithm.

5.1.3. Tree-based feature selection

Tree-based models have an inherent feature importance with which the best features, according to the feature importance, can be selected [**55**]. Note that features selected by an algorithm like the tree-based feature selection, is not necessarily the best features for e.g. the SVM.

5.2. The machine learning algorithms

In this study the Python module scikit-learn has been used which implements different machine learning algorithms and the main base algorithm used was SVM, which was explained in greater detail in Section 2.3.1. Scikit-learn offer different implementations of SVMs and depending on implementation varying performances was achieved, both regarding accuracies during classification and regarding fit-time complexity, i.e. the time it takes to train a model, and also the time for carry out predictions on new data. Scikit-learns implementations of SVMs are based on either libsvm or liblinear which are libraries that implements tools for classification. However, these libraries will not be further discussed but more information about them can be found in [**56**] and [**57**]. The following sub-sections discuss the different implementations of SVM that are present in Scikit-learn and which were used in this study.

5.2.1. SVC

This implementation of SVM, which documentation can be found in [**24**], offers the choice to apply one of several predefined kernel functions or the ability to apply an own implementation of a kernel function. Depending on which kernel function that is chosen different parameters are of interest. However, in this study the kernel *Radial Basis Function* (RBF) was the only kernel function that was used. Regarding the RBF kernel, the parameters of interests are C and gamma. A drawback of using SVM for classification is that the training phase is not easily scalable with more than a couple of 10 000 samples. According to the documentation the fit time complexity is more than quadratic with the number of samples [**24**].

5.2.1.1 C

There is one parameter that is common to all of the kernel functions, namely the parameter *C* [**58**]. The *C* parameter controls the tradeoff between the simplicity of the decision surface and the misclassification of the training examples. A low *C* yields a smooth decision surface. While with a high *C* the model tries to classify all of the training examples correctly, the complexity is introduced by giving the model more support vectors as training examples. For example, if the data is noisy a low value of *C* may be a better choice since we do not want the model to fit with respect to the noisy observations. If a too high a value of C is used then overfitting may occur.

5.2.1.2 gamma

When using RBF as the kernel function the parameter *gamma* can be tuned in order to define how far the influence of a single support vector reaches [**58**]. Think of the support vector to have a radius of a certain size attached to it, which is tuned by the parameter gamma. When using the trained model to predict on new data and an observation "lands" within a support vectors radius, it is then classified as belonging to the same class as that support vector. With a low value of gamma a support vectors influence reaches further out compared to a high value which causes its influence to be smaller. This implies that the parameter gamma can be seen as the inverse of the radius of influence for the models support vectors. Something to keep in mind when tuning the gamma parameter is that if a too high a value of gamma is used overfitting can occur.

5.2.2. Linear SVC

This implementation of linear SVM is similar to the implementation discussed above with the kernel function set to *linear* [**59**]. However, this implementation is based on liblinear rather than libsvm and this implementation offers more choices in terms of penalties and loss functions. Also this implementation should scale a lot better with a larger number of samples compared to the SVC implementation previously discussed. There are some parameters that can be tuned in order to improve the performance. One of those parameters is *C* which has the same definition as in the SVC implementation discussed above.

In addition to the parameter C the two parameters *loss* and *penalty* can be tuned. However, after some initial testing it was seen that these two parameters did not have any particular impact on the performance of the model. So it was decided that the *loss* and *penalty* parameters were used with their default values.

6. Results

In this chapter results for 1 of the 6 experiments conducted will be presented in detail and the additional experiments will be summarized (the details about the additional experiments are presented in Appendix A). In each experiment where Test 1 describes a test in which all features were used, the test may be seen as a benchmark test, with which the succeeding tests, in that experiment, may be compared to. All the results that are presented are the approximate results for the models achieved by 10-fold cross-validation.

6.1. Heat map

In the experiments with the SVC implementation the best combination of the parameters C and gamma was decided by a grid search. A grid search simply means that all the combinations of the values of interest regarding the parameters are tested and this may be illustrated in a *heat map*. In Figure 18 a heat map is illustrated and as can be seen the heat map consist of two parts; (i) the green part, which is the actual result achieved during the grid search, and (ii) the red part, which indicates which value a specific color in the grid search corresponds to. As an example, in Figure 18 the heat map shows the overall accuracy achieved while varying the parameters C and gamma. By inspecting the column (the red part) that maps a color to a value of accuracy it can be seen that the best combination of C and gamma is 100 and 0.1 respectively.

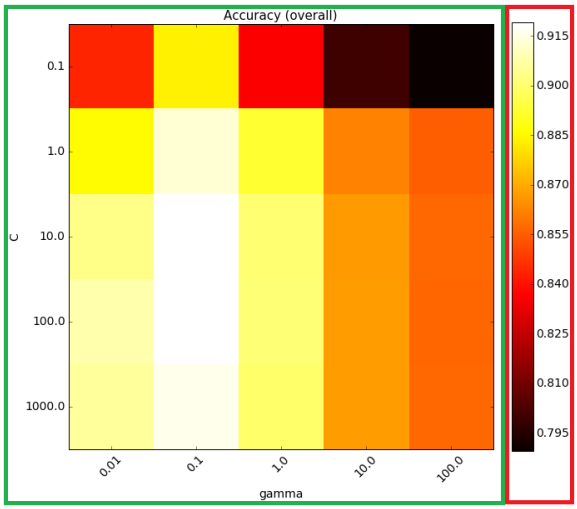

Figure 18 Result of a grid search for the parameters C and gamma illustrated in a heat map.

6.2. Time Complexity Regarding SVC with RBF Kernel

The SVC implementation is based on libsvm and as mentioned earlier according to the documentation regarding the SVC classifier the fit time complexity scales badly with increasingly number of samples. The following experiments show how the SVM with RBF kernel scales in relationship to the number of samples used during training and testing and also in relationship to the number of features used. The results also show the fit time complexity while varying the C and gamma parameters, including the influence the number of test samples has during the training of the model.

As can be seen in Figure 19 the time during the training of the model increases rapidly with the number of samples. Already at around 25 000 samples (note that 80% of the samples are used during the training) the training takes around 60 seconds and from this point forward it just gets worse. Fortunately, as Figure 20 shows it seems that the accuracy of the model does not increase very much with the increase of number of samples used during the training of the model. To use around 10 000 to 20 000 samples during the training seems to suffice in order to give an overall estimate of the performance of the model.

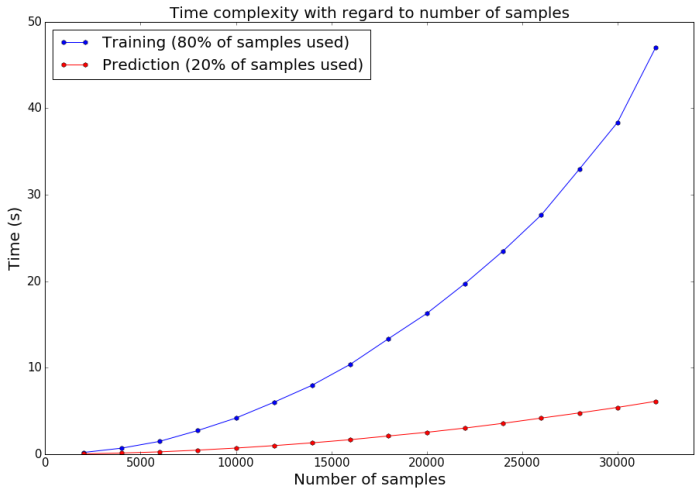

Figure 19 Time complexities in relationship to the total number of samples used during training and evaluation.

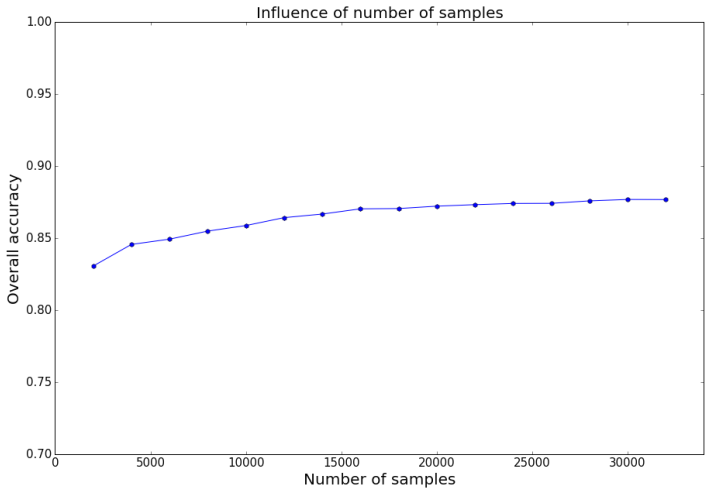

Figure 20 The influence on the overall accuracy with increasing number of training samples.

In Figure 21 and Figure 22 the time complexity with respect to the SVC parameters C and gamma are shown. When C grows large the time it takes training the model grows large, this behavior can be explained by the model trying to find more and more support vectors, i.e. less error in the training data, during the training phase. In contrast, the range of values chosen for gamma seems not to affect the time during training as much as the parameter C does. With regard to these test results in addition to the purpose of C and gamma, recall Section 5.2.1, a decision was made that a logarithmic range from 10^{-1} to 10^{3} for C and a logarithmic range from 10^{-2} to 10^{2} gamma would suffice as a starting point. If the best parameters are at an edge of the grid a subsequent search could be carried out.

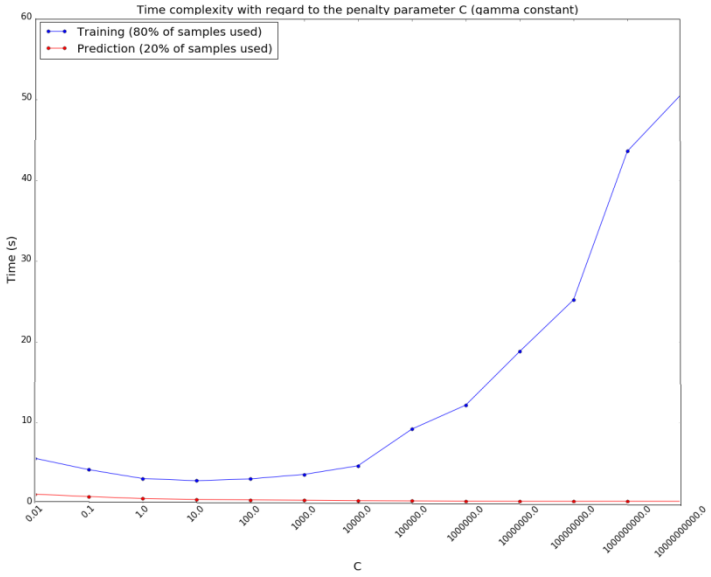

Figure 21 Time complexities for both training and prediction in relationship to the value of C.

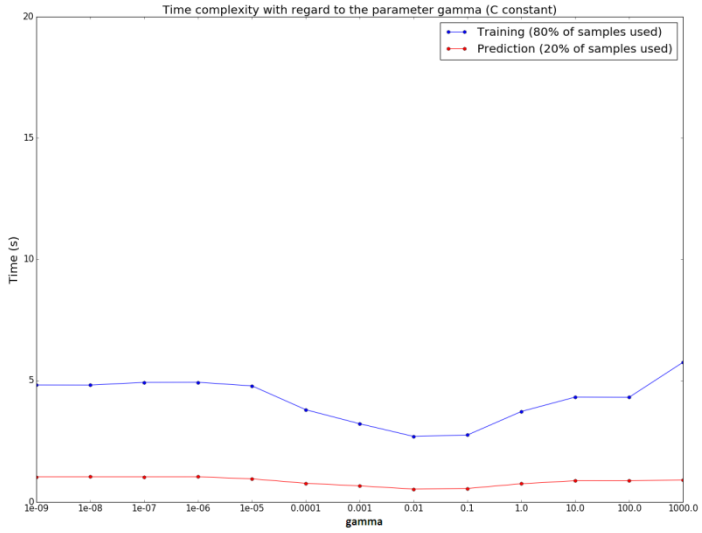

Figure 22 Time complexities for both training and prediction in relationship to the value of gamma.

There has been a brief discussion on feature selection in Section 2.2.3 and as can be seen in Figure 23 the time for training a model increases at a steady pace with the number of features considered and the prediction of new observation also increases, but not as much. In time critical systems decreases of the prediction time as much as possible is of importance and thus reduce the number of features used. Another important aspect regarding the number of features is that the features need to be extracted and/or calculated from the traffic data before predictions can be

carried out and the reduction of features can greatly reduce these extraction and calculation times in the system, so the reduction of features is twofold.

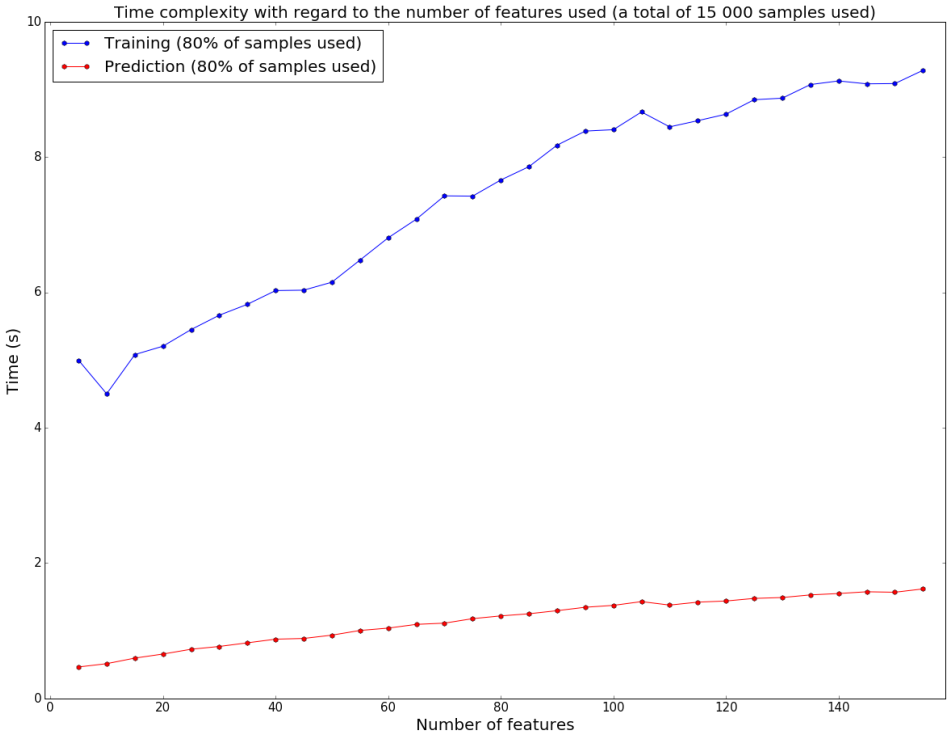

Figure 23 The time complexities for both training and prediction in relationship to the number of features used. A total of 15 000 samples were used during these calculations.

6.3. Time Complexity Regarding Linear SVC

In contrast to the SVC implementation the Linear SVC is implemented based on liblinear and according to the documentation in [**59**] the Linear SVC implementation should scale better with the increase in number of samples than the SVC implementation. And as can be seen in Figure 24 it seems that the time for training the models increases about linearly to the number of samples. The time for predictions to be carried out does not seem to increase much with the increase of samples, if at all. In comparison to the SVC the Linear SVC is the clear choice to use if time is of the essence and if the data is linearly separable the performance regarding prediction accuracy should be equal to the SVC.

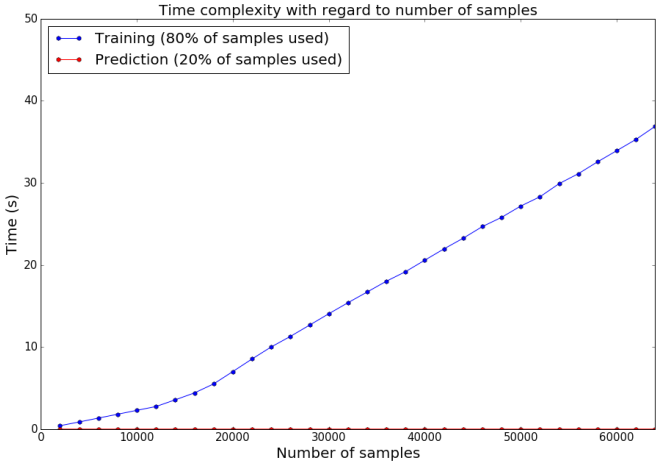

Figure 24 Time complexities in relationship to the total number of samples used during training and evaluation.

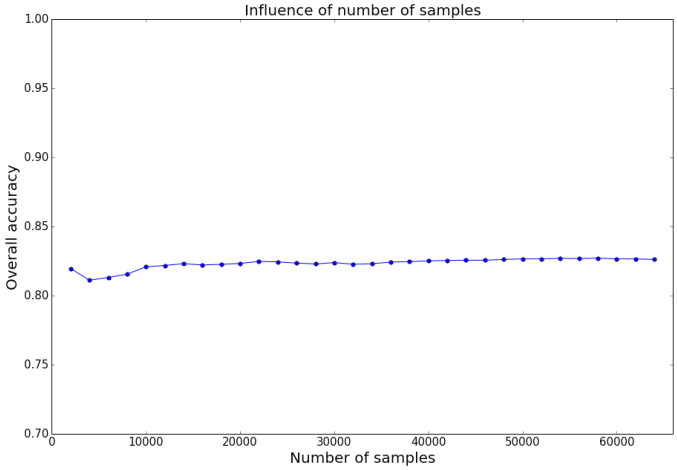

Figure 25 The influence on the overall accuracy with increasing number of training samples.

6.4. Experiment 1 - SVC with RBF kernel using packet feature set: 20 first packets

Algorithm: SVC with RBF kernel

Initial feature set: Packet feature set

The features were computed based on the first 20 packets. Features were computed at packet indices 6, 8, 10, 12, 14, 16, 18 and 20.

Sampled data set: *10k_10k*

Filtering specifics: The payload size was defined as >70 bytes. Only traffic flows with at least 10 packets were considered.

Filtered data set:

```
Number of Unique Flows   :  16670
Number of Unique Services:  129

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |      7617 | 45.69
HTTP                                    |      2260 | 13.56
BitTorrent transfer                     |       880 | 5.28
uTP                                     |       769 | 4.61
Youku                                   |       700 | 4.20
MiBox                                   |       678 | 4.07
DNS                                     |       422 | 2.53
BitTorrent KRPC                         |       408 | 2.45
RTMP                                    |       299 | 1.79
PPStream                                |       285 | 1.71

Freq Video

SERVICE   |Count      | Share (%)
_____________________________________
Video     |     10359 | 62.14
Other     |      6311 | 37.86
```

6.4.1. Test 1 - All features

In Test 1 all of the features in the initial feature set were used which constituted a total of 158 features. All the features were used in order to get a general understating of how well the classifier could perform. Test 1 can be seen as the benchmark test which all other tests with fewer features can be compared to. The parameter C was set to 100 and the parameter gamma was set to 0.01 by inspection of Figure 26 and Figure 27. The corresponding classifiers performance can be seen in the confusion matrix in Table 2. An overall accuracy of 91.53% was achieved and the classifier reached a recall of 93.59% and a precision of 92.86% regarding Video.

Feature selection: No feature selection.

Features: All features in the initial feature set, a total of 158 features.

Table 2 Exp. 1 Test 1 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	5548	747	6295	88.13%	89.30%	
	Video	665	9710	10375	93.59%	92.86%	
	Total	6213	10457	16670			91.53%

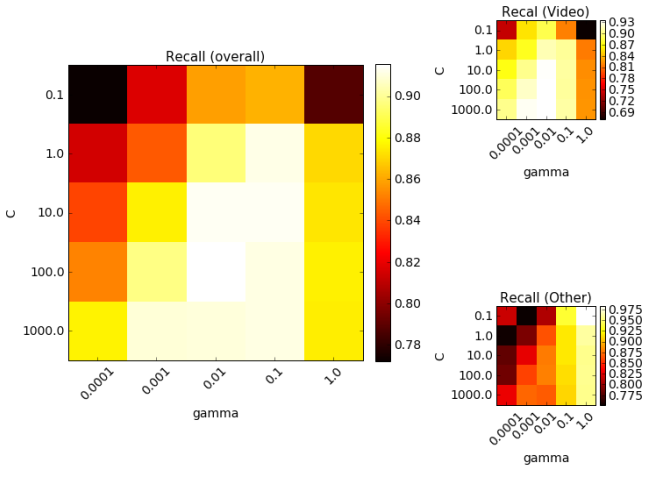

Figure 26 Exp. 1 Test 1 recall while varying C and gamma.

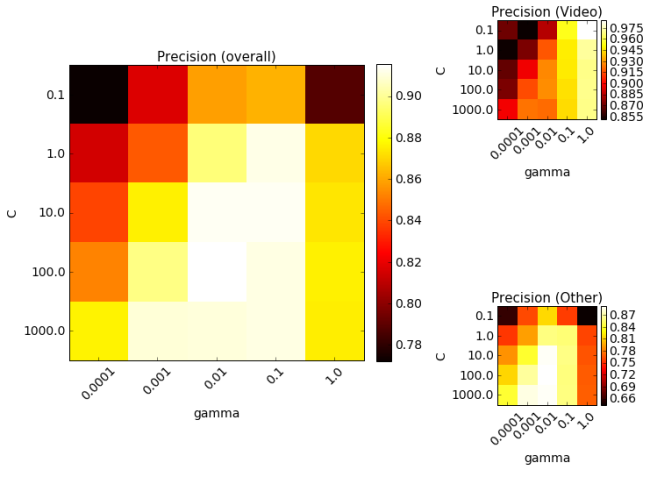

Figure 27 Exp. 1 Test 1 precision while varying C and gamma.

6.4.2. Test 2 – Features based on the first 20 packets

In Test 2 a total of 15 features were used which were all of the features computed for the twentieth packet, i.e. the features which were computed based on the first 20 packets in addition to the inter arrival time for the last packet. These features were picked to see how well the classifier could perform when only considering the maximum number of packets, i.e. 20 packets in this case. The best values for C and gamma were chosen according to the heat maps in Figure 28 and Figure 29 to be 1000 and 10 respectively and the results achieved for the corresponding classifier can be seen in the confusion matrix in Table 3. An overall accuracy of 88.10% was achieved which may seem a little low compared to the benchmark test, i.e. Test 1 - *All features*. If Video is considered the FNs only slightly increased compared to the benchmark test while the FPs increased by almost 2/3rds and as a result the recall only slightly decreased, a recall of 92.69% was achieved, while the precision got worse, only 88.69% was achieved. In addition, as a result the recall regarding Other decreased significantly and landed on 80.52%.

Feature selection: All features computed for the twentieth packet.

Features: 15 features total.

Feature set = {BytesDwUp-20, BytesDw-20, BytesDwPayl-20, NrDownPkts-20, NrDownPaylPkts-20, PosSumDwnPkts-20, PosSumDwnPaylPkts-20, PosSumDwnPktsEven-20, PosSumDwnPaylPktsEven-20, StdDevDwnPkts-20, StdDevDwnPaylPkts-20, DwnPaylRun-20, DwnPaylRun200-20, IATUp-20, IATDwn-20}

Table 3 Exp. 1 Test 2 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	5069	1226	6295	80.52%	86.99%	
	Video	758	9617	10375	92.69%	88.69%	
	Total	5827	10843	16670			88.10%

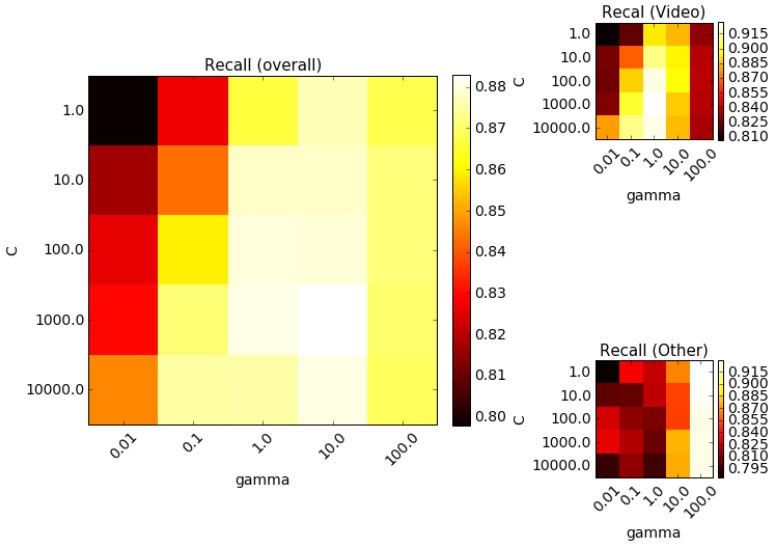

Figure 28 Exp. 1 Test 2 recall while varying C and gamma.

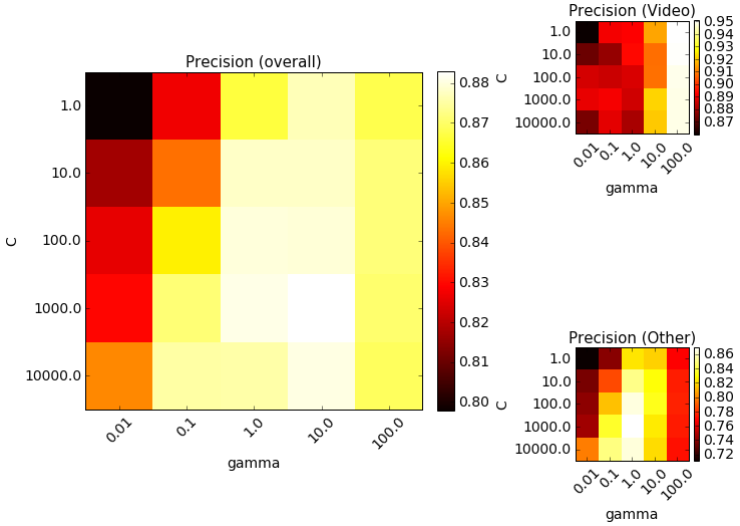

Figure 29 Exp. 1 Test 2 precision while varying C and gamma.

6.4.3. Test 3 – Single feature

In Test 3 only 1 feature was selected and it was selected because the Decision Tree model, which has an inherent feature importance ranking, ranked it far above every other feature. 1 feature was used in order to explore the very extreme case; fewer features result in a less complex system (both regarding the prediction and training of a classifier but also in the aspect that the features need to be computed). The parameters C and gamma were chosen as 100 and 1 respectively according to the heat maps in Figure 30 and Figure 31. The performance for the corresponding classifier can be seen in the confusion matrix in Table 4. The model only achieved an overall accuracy of 78.81% so it seems that 1 feature is insufficient.

Feature selection: The most important feature according to the Decision Tree model.

Features: 1 feature in total.

Feature set = {DwnPaylRun200-20}

Table 4 Exp. 1 Test 3 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	4362	1933	6295	69.29%	73.18%	
	Video	1599	8776	10375	84.59%	81.95%	
	Total	5961	10709	16670			78.81%

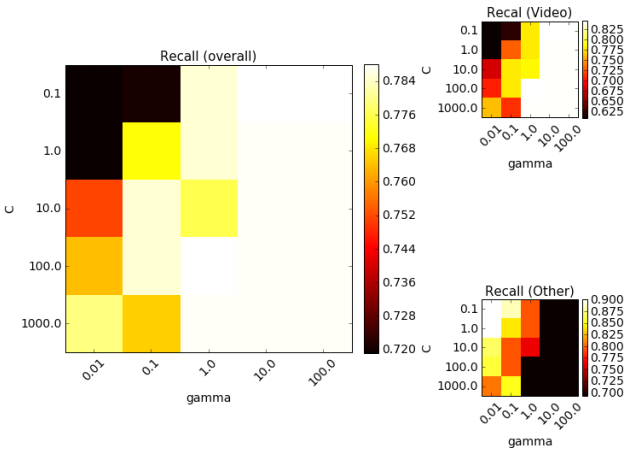

Figure 30 Exp. 1 Test 3 recall while varying C and gamma.

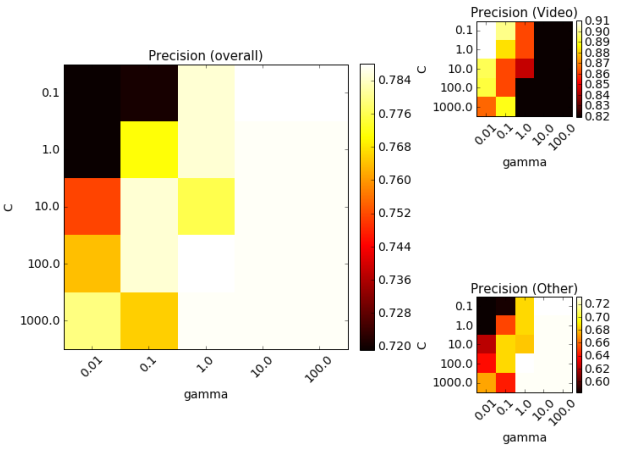

Figure 31 Exp. 1 Test 3 precision while varying C and gamma.

6.4.4. Test 4 – All downward payload run features

In Test 4 all the downward payload run features were selected constituting a total of 16 features in order to see how good the downward payload run features seemed to be. The parameters C and gamma were chosen as 10 and 1 respectively by inspection of the heat maps seen in Figure 32 and Figure 33. The confusion matrix in Table 5 shows the result for the corresponding classifier. Considering that 16 features were used and the classifier only achieved an overall accuracy of 81.59% the downward payload run features seems in general to be insufficient.

Feature selection: All of the downward payload run features.

Features: 16 features total.

Feature set = {DwnPaylRun-6, DwnPaylRun-8, DwnPaylRun-10, DwnPaylRun-12, DwnPaylRun-14, DwnPaylRun-16, DwnPaylRun-18, DwnPaylRun-20, DwnPaylRun200-6, DwnPaylRun200-8, DwnPaylRun200-10, DwnPaylRun200-12, DwnPaylRun200-14, DwnPaylRun200-16, DwnPaylRun200-18, DwnPaylRun200-20}

Table 5 Exp. 1 Test 4 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	4802	1493	6295	76.28%	75.29%	
	Video	1576	8799	10375	84.81%	85.49%	
	Total	6378	10292	16670			81.59%

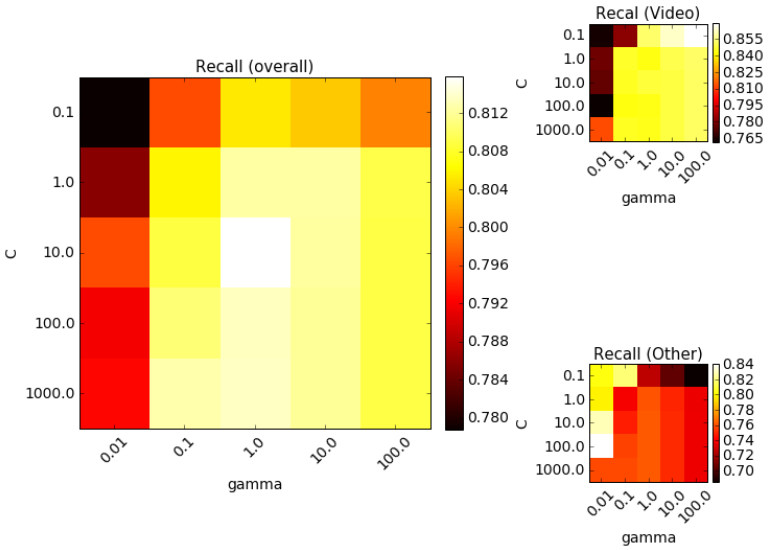

Figure 32 Exp. 1 Test 4 recall while varying C and gamma.

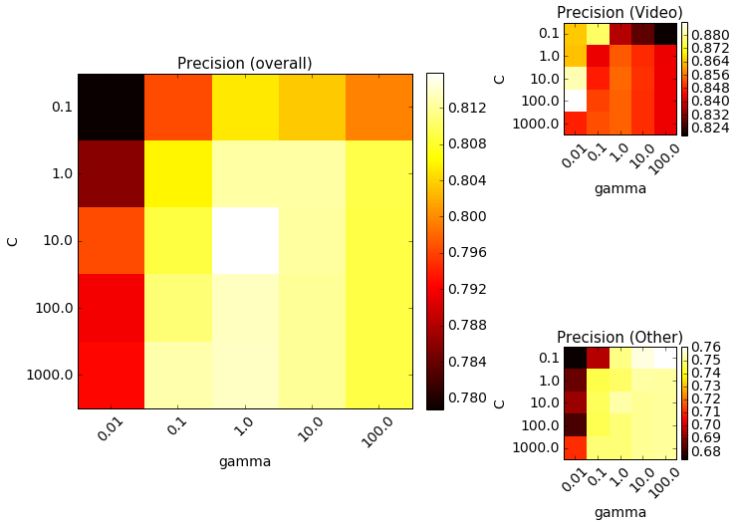

Figure 33 Exp. 1 Test 4 precision while varying C and gamma.

6.4.5. Test 5 – Payload size based features

In Test 5 13 features in total were used and the features were the payload sizes for the first 10 payload packets and the three statistical features average, variance and standard deviation based on the payload sizes. The payload size based features were chosen because in several of the previously made studies, briefly discussed in Chapter 3, payload size based features has been suggested to be effective. The parameters C and gamma were decided by inspection of the heat maps seen in Figure 34 and Figure 35 to 100 and 10 respectively. The performance for the corresponding classifier can be seen in the confusion matrix in Table 6. An overall accuracy of 90.40% was achieved, in addition the classifier also achieved reasonable high recall for both Video and Other and high precision regarding Video. Compared to the benchmark test, i.e. Test 1 - *All features*, the result are promising, the precision regarding Video are even higher.

Feature selection: Payload sizes and statistical features based on the payload sizes.

Features: 13 features total.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylMean, PaylVar, PaylStd}

Table 6 Exp. 1 Test 5 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	5726	569	6295	90.96%	84.74%	
	Video	1031	9344	10375	90.06%	94.26%	
	Total	6757	9913	16670			90.40%

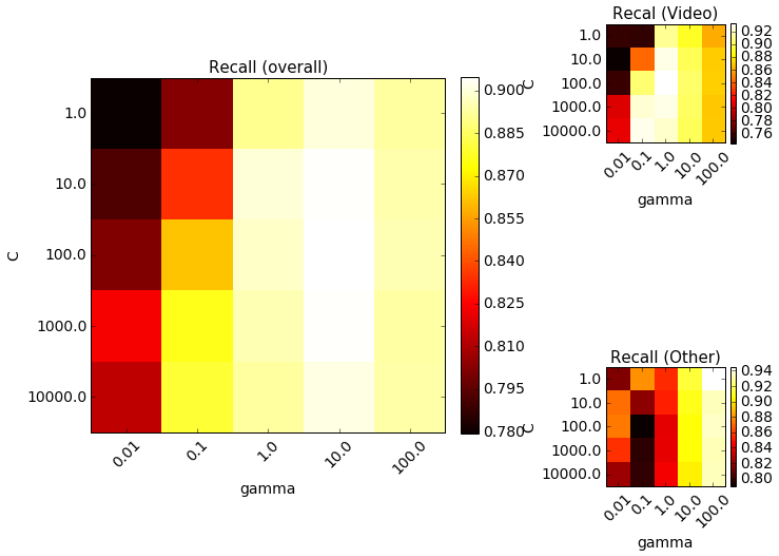

Figure 34 Exp. 1 Test 5 recall while varying C and gamma.

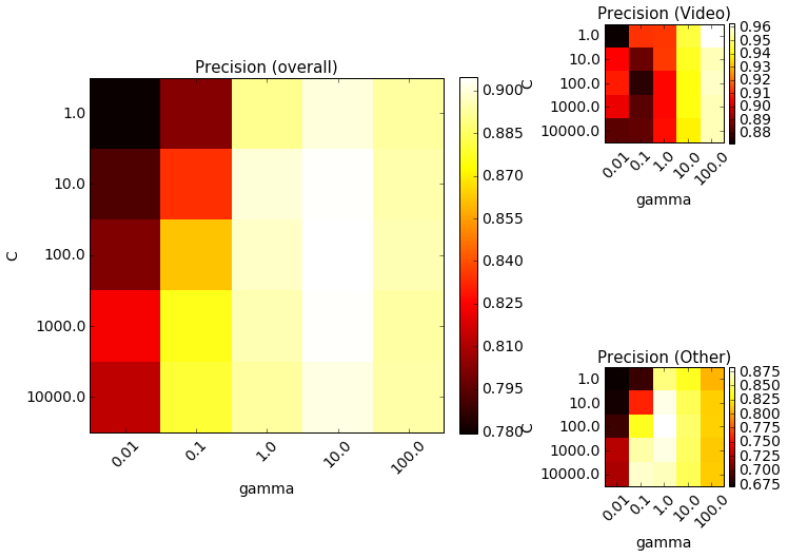

Figure 35 Exp. 1 Test 5 precision while varying C and gamma.

6.4.6. Test 6 – Payload based features

The features used in Test 6 were chosen to be based on payload packets in the downward direction adding, up to a total of 7 features. This combination of features was chosen in order to see how good the combination of the three statistical payload size based features, average, variance and standard deviation, in addition to the downward payload based features could perform. The values for C and gamma were chosen as 100 and 10 respectively by inspection of the heat maps in Figure 36 and Figure 37. The performance for the corresponding classifier can be seen in the confusion matrix in Table 7. An overall accuracy of 89.36% was achieved and in addition the recall and precision for Video is reasonable high. Considering that only 7 features were used the classifier performs well in comparison to the benchmark test, i.e. Test 1 - *All features*.

Feature selection:

Features: 7 features total.

Feature set = {PaylMean, PaylVar, PaylStd, DwnPaylRun200-20, BytesDwPayl-20, PosSumDwnPaylPkts-20, NrDownPaylPkts-20}

Table 7 Exp. 1 Test 6 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	5415	880	6295	86.02%	85.83%	
	Video	894	9481	10375	91.38%	91.51%	
	Total	6309	10361	16670			89.36%

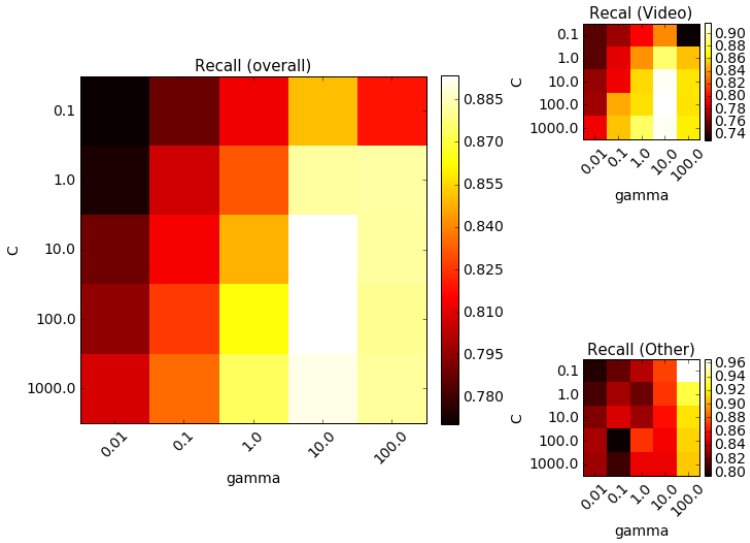

Figure 36 Exp. 1 Test 6 recall while varying C and gamma.

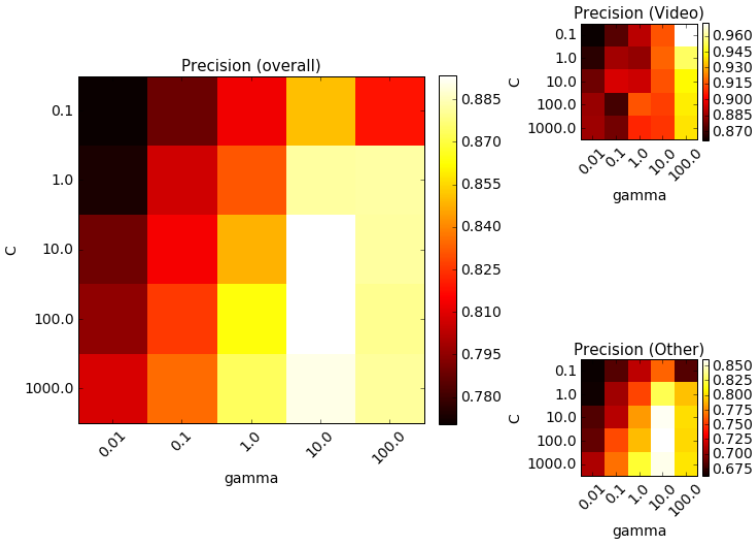

Figure 37 Exp. 1 Test 6 precision while varying C and gamma.

6.4.7. Test 7 – Position sum features

In Test 7 a total of 32 features were used which were all of the features regarding the position sum. This combination of features was chosen in order to see how good in general the position sum features seems to be. Both C and gamma were chosen as 1 by inspection of the heat maps in Figure 38 and Figure 39. The performance for the classifier can be seen in the confusion matrix in Table 8. Considering that 32 features were used the overall accuracy of 86.39% is deemed to be a bad score, however, the recall regarding Video is 90.90% which indicate that some of the position sum features could be useful in combination with other features.

Feature selection: All the position sum features.

Features: 32 features total.

Feature set = {PosSumDwnPkts-6, PosSumDwnPkts-8, PosSumDwnPkts-10, PosSumDwnPkts-12, PosSumDwnPkts-14, PosSumDwnPkts-16, PosSumDwnPkts-18, PosSumDwnPkts-20, PosSumDwnPaylPkts-6, PosSumDwnPaylPkts-8, PosSumDwnPaylPkts-10, PosSumDwnPaylPkts-12, PosSumDwnPaylPkts-14, PosSumDwnPaylPkts-16, PosSumDwnPaylPkts-18, PosSumDwnPaylPkts-20, PosSumDwnPktsEven-6, PosSumDwnPktsEven-8, PosSumDwnPktsEven-10, PosSumDwnPktsEven-12, PosSumDwnPktsEven-14, PosSumDwnPktsEven-16, PosSumDwnPktsEven-18, PosSumDwnPktsEven-20, PosSumDwnPaylPktsEven-6, PosSumDwnPaylPktsEven-8, PosSumDwnPaylPktsEven-10, PosSumDwnPaylPktsEven-12, PosSumDwnPaylPktsEven-14, PosSumDwnPaylPktsEven-16, PosSumDwnPaylPktsEven-18, PosSumDwnPaylPktsEven-20}

Table 8 Exp. 1 Test 7 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	4970	1325	6295	78.95%	84.04%	
	Video	944	9431	10375	90.90%	87.68%	
	Total	5914	10756	16670			86.39%

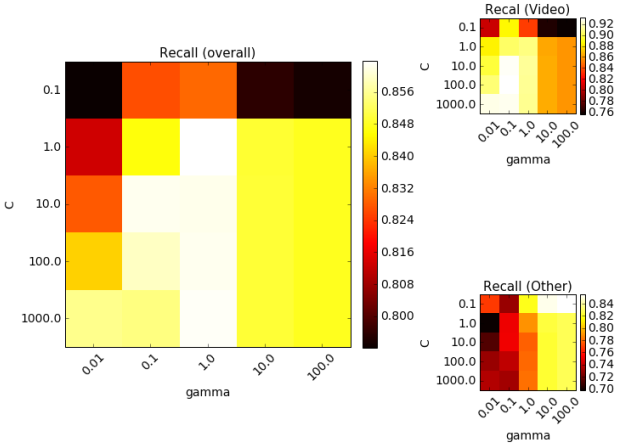

Figure 38 Exp. 1 Test 7 recall while varying C and gamma.

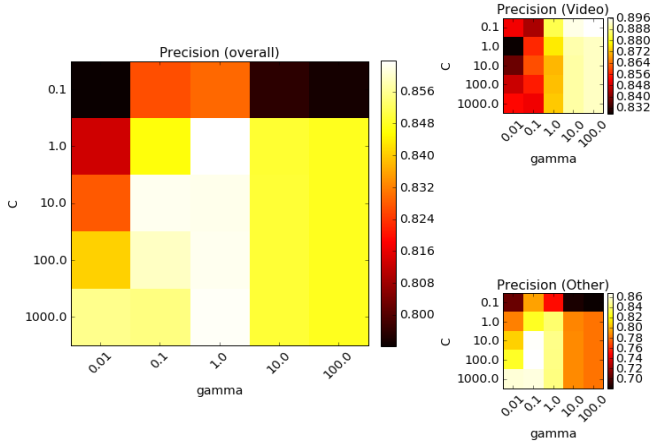

Figure 39 Exp. 1 Test 7 precision while varying C and gamma.

6.4.8. Test 8 – Inter arrival time features

For Test 8 all of the internal arrival time features were chosen, constituting a total of 40 features, in order to get a general understating of how good these features are. The best values for both C and gamma were arbitrary chosen as 10, the heat maps for the grid search can be seen in Figure 40 and Figure 41. The performance for the corresponding classifier can be seen in the confusion matrix in Table 9. The classifier only achieved an overall accuracy of 82.25% which is deemed to be a very bad score considering that 40 features were used.

Feature selection: All the IAT features.

Features: 40 features total.

Feature set = {IATUp-1, IATUp-2, IATUp-3, IATUp-4, IATUp-5, IATUp-6, IATUp-7, IATUp-8, IATUp-9, IATUp-10, IATUp-11, IATUp-12, IATUp-13, IATUp-14, IATUp-15, IATUp-16, IATUp-17, IATUp-18, IATUp-19, IATUp-20, IATDwn-1, IATDwn-2, IATDwn-3, IATDwn-4, IATDwn-5, IATDwn-6, IATDwn-7, IATDwn-8, IATDwn-9, IATDwn-10, IATDwn-11, IATDwn-12 ,IATDwn-13, IATDwn-14, IATDwn-15, IATDwn-16, IATDwn-17, IATDwn-18, IATDwn-19, IATDwn-20}

Table 9 Exp. 1 Test 8 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	4868	1427	6295	77.33%	76.06%	
	Video	1532	8843	10375	85.23%	86.11%	
	Total	6400	10270	16670			82.25%

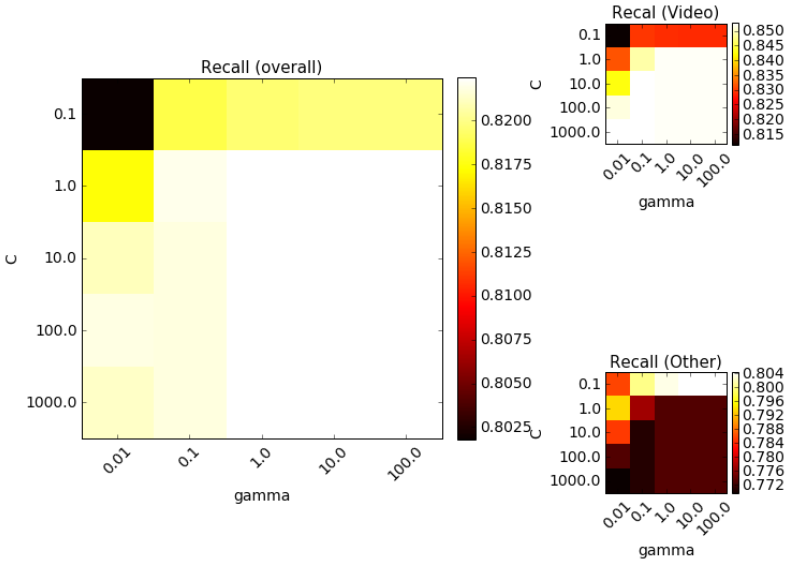

Figure 40 Exp. 1 Test 8 recall while varying C and gamma.

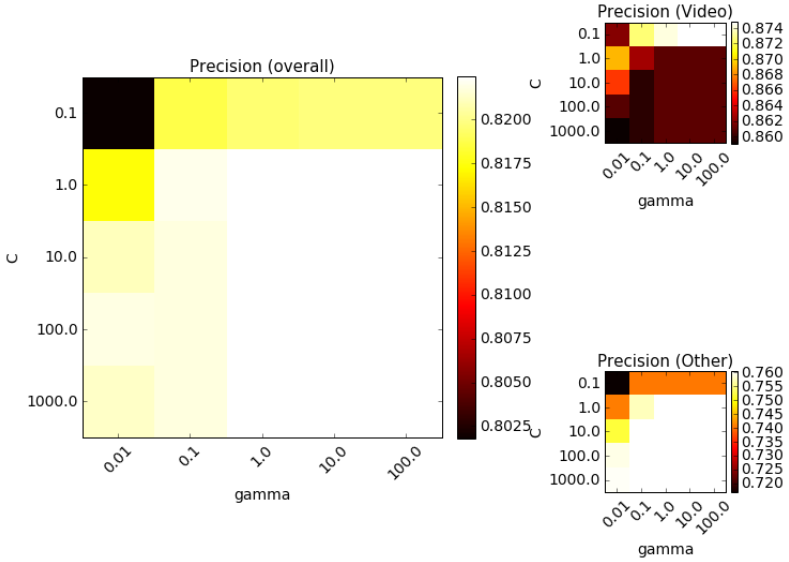

Figure 41 Exp. 1 Test 8 precision while varying C and gamma.

6.4.9. Experiment 1 analysis

The test results gathered during Experiment 1, in which a total of 8 different tests where different sets of features were used, are summarized in Figure 42, Figure 43, Figure 44 and Figure 45. If recall and precision are examined first one can see that the overall trend throughout the tests is that "Video" seems to be the class of traffic that is more easily classified. The reason for this could be the combination that the test data used was composed of roughly 2/3rds of "Video" traffic and 1/3rd "Other" in addition to the fact that there are a lot more Internet services in the "Other" class than in the "Video" class. If the data set that was used is examined one can see that there is an unevenly representation of the different services which implies that some services might be underrepresented during the training of the model which may lead to worse performance. However, if the characteristics between services in one class look the same, or close to, there should not be a problem if some services are underrepresented.

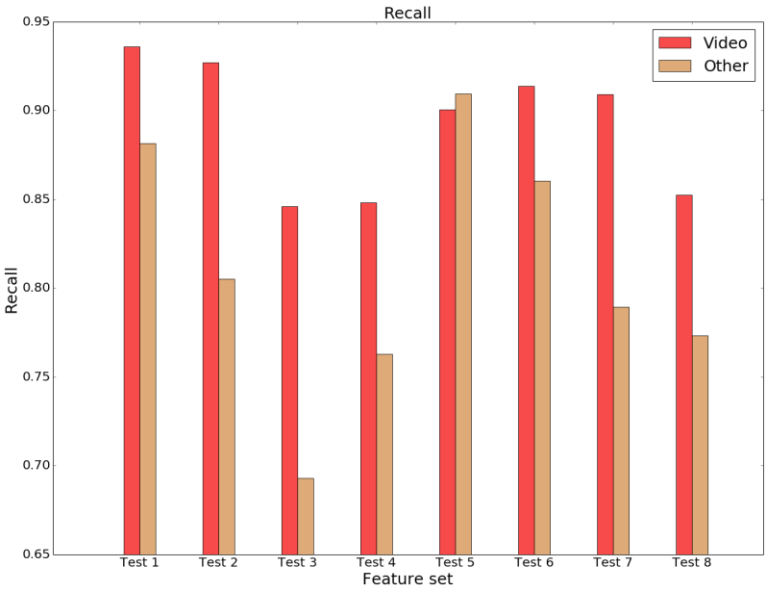

Figure 42 Exp. 1 Recall for Test 1 through 8.

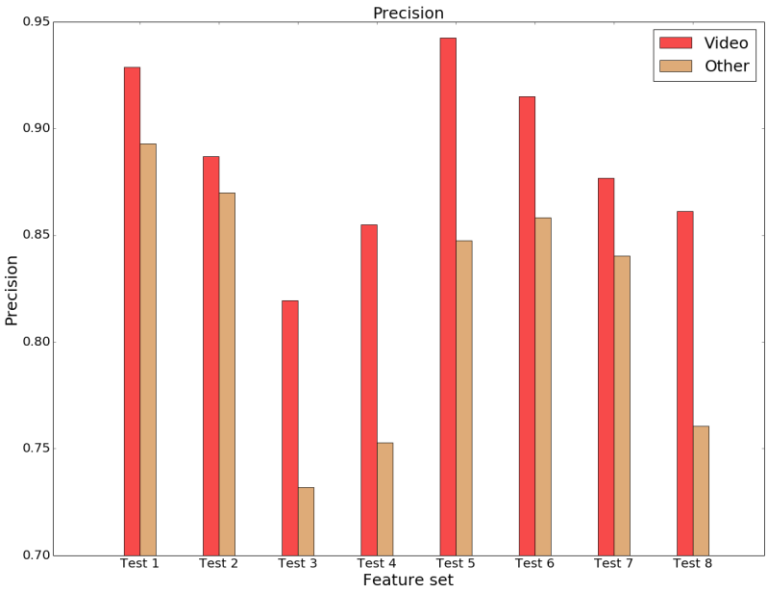

Figure 43 Exp. 1 Precision for Test 1 through 8

The highest overall accuracy of 91.53% was achieved by *Test 1 - All features* in which all of the features in the initial feature set were included, i.e. 158 features in total. Test 1 also had the best ROC curve including an AUC score of 0.96. The lowest overall accuracy of 78.81% was achieved by *Test 3 – Single feature* which only had 1 feature namely the payload run feature based on the 20 first packets. In addition, Test 3 also achieved really low recall and precision

when considering the "Other" class. If *Test 5* – Payload size based features is examined, which only uses payload sizes for the first 10 packets and three statistical features based on these payload sizes, one can see that it achieved an accuracy of 90.40%. Test 5 also achieved a precision of 94.26% considering "Video" but not as good considering "Other". However, Test 5 achieved a recall above 90% for both classes which no other classifier did. Furthermore, Test 5 had almost as good a ROC curve as Test 1 with an AUC score of 0.94. The *Test 6* – Payload based features closely follows with an overall accuracy of 89.36%, and considering that only 7 features were used, despite a little worse performance regarding recall and precision, its results are reasonably good in comparison to the benchmark test, i.e. *Test 1 - All features,* and Test 5.

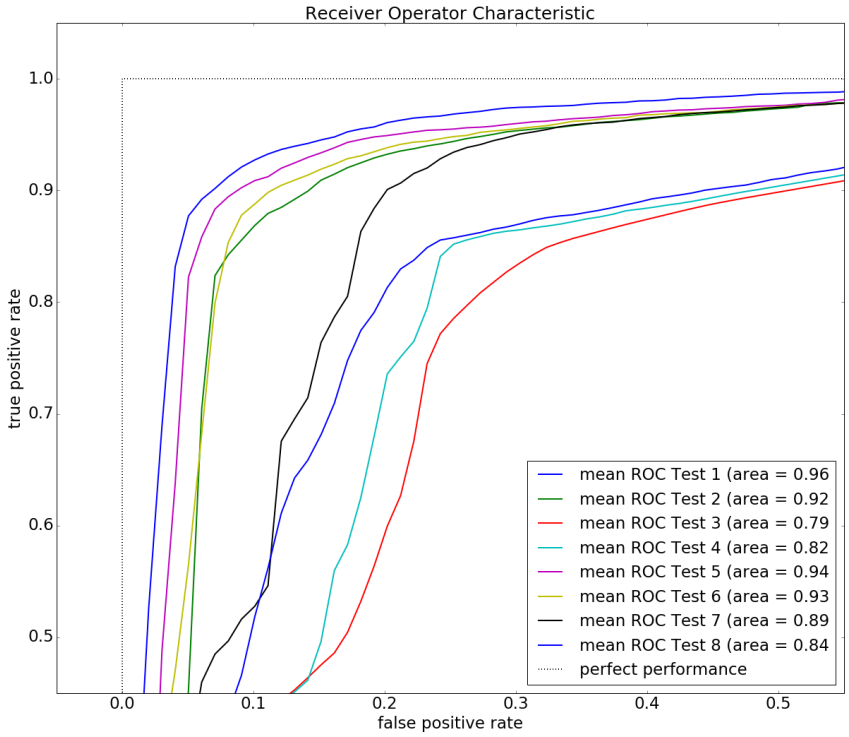

Figure 44 Exp. 1 Comparison of ROC curves for Test 1 through 8.

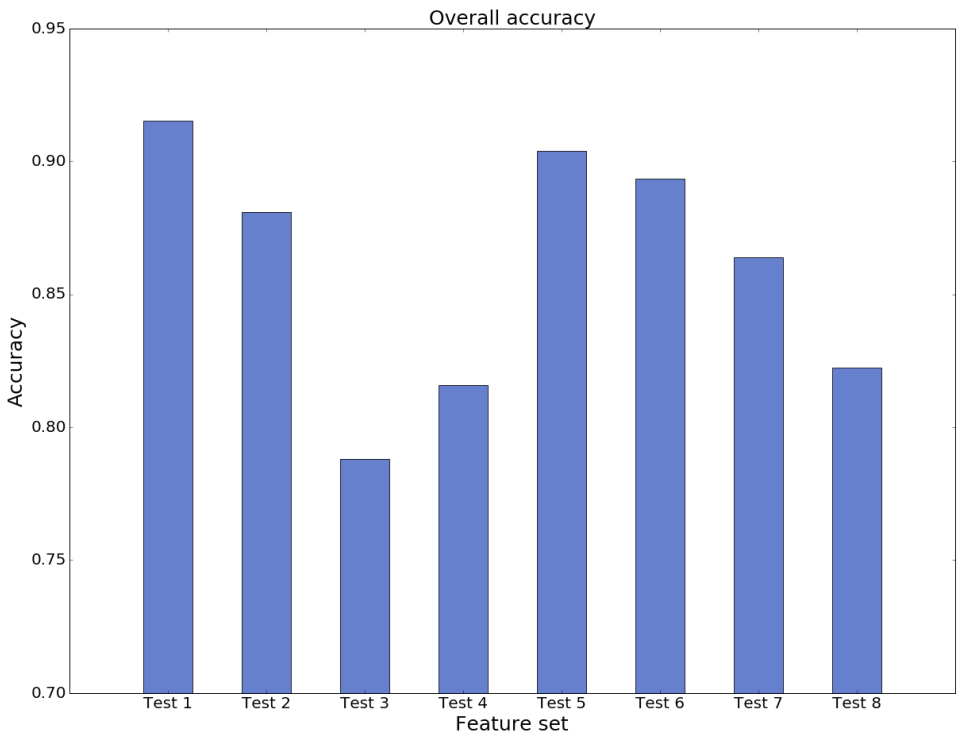

Figure 45 Exp. 1 Overall accuracy for test 1 through 8.

6.4.10. Experiment 1: Results parallel study – Decision Tree and Random Forest

As mentioned earlier, a parallel study was carried out in which Decision Tree and the Random Forest algorithms were used [**7**]. The sampled dataset that was used was 100k_100k so the number of training samples is not the same as for the tests with the SVMs classifiers, however, the overall proportion between the services should have been the same. The result for the Decision Tree classifier is summarized in Table 10. An overall accuracy of 94.2% was achieved which can be compared to *Test 5* – Payload size based features where an overall accuracy of 90.40% was achieved which is a lot worse and the same goes for recall and precision considering Video. However, as mentioned the Decision Tree classifier used more training samples which may increase the scores slightly.

Table 10 Performance metrics for Decision Tree classifier.

	Recall	Precision	Overall accuracy
Other	91.9%	92.7%	94.2%
Video	95.6%	95.1%	

In Table 11 the result for the Random Forest classifier is summarized. And as can be seen the Random Forest classifier outperforms the Decision Tree classifier, which is expected since a Random Forest is composed of several Decision Trees, recall Section 2.3.3. An overall accuracy of 96.5% was achieved and recall and precision regarding Video even slightly higher. According to these results it seems that the Random Forest classifier is by far the best classifier considered in this study.

Table 11 Performance metrics for Random Forest classifier.

	Recall	Precision	Overall accuracy
Other	96.0%	94.8%	96.5%
Video	96.8%	96.8%	

6.5. Experiment 2

Experiment 2 was executed in order to see how well linear SVMs could perform. The results may be found in Appendix A - *Experiment 2 - Linear SVC with packet feature set: 20 first packets.* The hypothesis was that the data are nonlinear separable. The overall accuracy achieved when all the 158 features were used was 84.59% which is not a terrible score, however, the performance is most likely to decrease when fewer features are used. And according to these results a linear SVM does not seem to suffice to accurately classify the Internet traffic data. Thus not much testing regarding the linear SVM was executed.

6.6. Experiment 3

In Experiment 3 the payload feature set was used which means that only payload packets were considered. The results for Experiment 3 is found in Appendix A - *Experiment 3 - SVC with RBF kernel using payload feature set : 10 first payload packets*. This experiment was executed to see how well the payload feature set implementation worked. An overall accuracy of almost 92% was achieved and the recall and precision regarding Video was high, 93.67% and 96.61% respectively. However, the recall and precision regarding Other was worse. No more testing with the payload feature set was conducted since it does not take all of the packets in a flow into consideration; a decision was made to focus on the packet feature set instead. However, it would still be interesting to see the performances of classifiers with a different sampling of the data set. Note that during this experiment only ~16% of the traffic flows belonged to the Other class, better performances could potentially be achieved but this was not achieved in the timeframe of this study.

6.7. Experiment 4

The purpose of Experiment 4 was to examine whether the models could perform better when considering a larger range of packets in comparison to *Experiment 1 - SVC with RBF kernel using packet feature set: 20 first packets* which is found in Section 6.4. The hypothesis was that when a larger range of packets are considered a classifier should achieve better performances, except from a time consuming perspective. In Experiment 4 an assumption was made that all type of video flows are larger than 50 packets, i.e. only flows with at least 50 packets were considered. Experiment 4 reflects *Experiment 1 - SVC with RBF kernel using packet feature set: 20 first packets* in the sense that the same kinds of features were used relative to the number of packets considered. However, there were some differences between the Experiments; (i) the sampled data set, (ii) as mentioned the number of packets considered, (iii) Experiment 4 had a larger range for which the features were computed and lastly (iv) the inter arrival time features were excluded.

The detailed results regarding Experiment 4 can be found in Appendix A - *Experiment 4 – SVC with RBF kernel using packet feature set: 50 first packets* whereas the summary of the results can be found in Table 12. And as can be seen by these results there seems to be no

improvement to Experiment 1, rather a decline. The reason that the performances do not improve is likely to the fact that the traffic flows the classifiers tried to predict had become harder to discriminate. This can be explained by the assumption that video traffic flows has more than 50 packets and as a result this data set contains more "hard to classify" traffic flows. For example the proportion of HTTP traffic flows went from ~14% to ~25% after the filtration. HTTP traffic is believed to be hard to classify since web traffic can be of a wide variety of different content, which could include some video traffic as well that has not explicitly been given a Video service.

6.8. Experiment 5

In Experiment 5 all of the asymmetric flows were removed, i.e. flows that only seem to contain packets in one direction, in order to see how the asymmetric flows impact the classifiers performances. The hypothesis was that the classifiers performances would improve on all fronts. Experiment 5 also reflects *Experiment 1 - SVC with RBF kernel using packet feature set: 20 first packets* in the sense that the same kinds of features were used and the features were based on the same number of packets. However there were some differences; (i) the sampled data set, (ii) as mentioned, the asymmetric flows were removed and lastly (iii) the inter arrival time features were excluded.

According to the results, which can be found in Appendix A - *Experiment 5 – Asymmetric flows removed SVC with RBF kernel using packet feature set: 20 first packets*, the asymmetric flows seems to have a bad impact on the performance. This implies that if asymmetric flows are of interest to classify then a separate classifier that handles asymmetric flows should be trained.

6.9. Experiment 6

Experiment 6 was an attempt to see if there may be a better way of dividing the different Internet services instead of clustering all of the services considered as video applications into the same class. As a first attempt the service "HTTP media stream" is considered to be its own class (note that in the results this class is still called Video). The results are found in Appendix A - Experiment 6 – *Alternative Video grouping SVC with RBF kernel using packet feature set: 20 first packets*.

The overall accuracy was ~95% for the three tests conducted and the precision regarding Video is above 95% for all three tests while the recall for Video is worse. However, the results show that another grouping of the different services could be of interest and should not be totally ignored. For example video traffic could be divided into “Live video streaming” and “Video streaming” or each service could be its own class even. No further testing by dividing the Internet traffic into different groups was done due to time constraints.

6.10. Summary of results

In Table 12 a summary of all the experiments may be found. The overall accuracy together with the recall and precision regarding Video are shown. As can be seen the overall best performance for each experiment is achieved when all of the features considered are used, which was expected. Apart from the tests where all features are used, more interesting is that the tests where a reduced number of features are used.

6.10.1. Payload size based features

The tests where the feature set includes the individual payload size features in addition to the statistical features based on the payload sizes seems to perform good overall if compared to the tests where all features are used. And in rough numbers the number of features is reduced to ~1/10th of the initial number of features.

6.10.2. Payload based features

The tests where the feature set includes the payload based features alone closely follows the tests where the payload size based features were used. Furthermore, the number of features is less, only 7 features were used in those tests.

6.10.3. Single feature

The tests where 1 feature alone is used performs the worst without exception.

Table 12 Summary of the experiments.

Exp. X / Test Y- Approach	Data set	Filtration	Initial feature set/based on N first packets	Nr. Of features	Feature description	Recall Video	Prec. Video	Overall acc.
Exp. 1/1 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	158	All	93.59%	92.86%	91.53%
Exp. 1/2 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	15	Features computed for the 20 first pkts.	92.69%	88.69%	88.10%
Exp. 1/3 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	1	Highest ranked feature according to Decision Tree	84.59%	81.95%	78.81%
Exp. 1/4 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	16	All the downward payl. run features	84.81%	85.49%	81.59%
Exp. 1/5 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	13	10 first individual payl. sizes and 3 statistical features	90.06%	94.26%	90.40%
Exp. 1/6 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	7	Payl. based features	91.38%	91.51%	89.36%
Exp. 1/7 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	32	All the position sum features	90.90%	87.68%	86.39%
Exp. 1/8 SVC RBF	10k_10k	Flows < 10 pkts. removed	Pkt./20	40	All the inter arrival time features	85.23%	86.11%	82.25%
Exp. 2/1 Lin. SVC	10k_10k	Flows < 10 pkts. removed	Pkt./20	158	All	88.29%	87.12%	84.59%
Exp. 2/2 Lin. SVC	10k_10k	Flows < 10 pkts. removed	Pkt./20	13	10 first individual payl. sizes and 3 statistical features	75.58%	83.91%	75.78%
Exp. 3/1 SVC RBF	10k_10k	Flows < 10 pkts. removed	Payl./10	35	Feature selection by variance	93.67%	96.61%	91.97%
Exp. 4/1 SVC RBF	100k_100k	Flows < 50 pkts. removed	Pkt./50	172	All	91.20%	92.43%	91.90%
Exp. 4/2 SVC RBF	100k_100k	Flows < 50 pkts. removed	Pkt./50	13	Features computed for the 50 first pkts.	88.26%	88.72%	88.57%
Exp. 4/3 SVC RBF	100k_100k	Flows < 50 pkts. removed	Pkt./50	1	Highest ranked feature according to Decision Tree	74.44%	83.02%	79.70%
Exp. 4/4 SVC RBF	100k_100k	Flows < 50 pkts. removed	Pkt./50	22	All the downward payl. run features	77.77%	85.80%	82.53%
Exp. 4/5	100k_100k	Flows < 50 pkts.	Pkt./50	28	25 first individual payl. sizes and	86.92%	95.46%	91.43%

SVC RBF		removed			3 statistical features			
Exp. 4/6 SVC RBF	100k_100k	Flows < 50 pkts. removed	Pkt./50	7	Payl. based features	85.53%	95.08%	90.60%
Exp. 4/7 SVC RBF	100k_100k	Flows < 50 pkts. removed	Pkt./50	44	All the position sum features	81.90%	83.65%	83.02%
Exp. 5/1 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed	Pkt./20	134	All	91.46%	93.52%	92.56%
Exp. 5/2 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed	Pkt./20	13	10 first individual payl. sizes and 3 statistical features	90.91%	94.01%	92.56%
Exp. 5/3 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed	Pkt./20	7	Payl. based features	87.14%	94.43%	91.00%
Exp. 5/4 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed	Pkt./20	17	10 first individual payl. sizes, 3 statistical and payl. based features	92.05%	92.97%	92.55%
Exp. 6/1 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed. Video=1 serv.	Pkt./20	10	10 first individual payl. sizes	88.46%	95.52%	94.95%
Exp. 6/2 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed. Video=1 serv.	Pkt./20	13	10 first individual payl. sizes and 3 statistical features	88.11%	95.67%	94.89%
Exp. 6/3 SVC RBF	100k_100k	Flows < 10 pkts. and asym. flows removed. Video=1 serv.	Pkt./20	17	10 first individual payl. sizes, 3 statistical and payl. based features	89.04%	95.89%	95.24%

7. Conclusions and Future Work

In this study the machine learning method SVM has been used and evaluated in order to explore whether it is possible to classify Internet traffic by characteristics extracted from the traffic flows, instead of using methods such as DPI. Two different approaches to the feature extraction has been used; one in which the features were based on the payload packets alone and one in which the features were based on all kinds of packets, where the latter of the two has been in focus. In addition different filtrations and samplings of the data set has been used as basis for the feature sets: Feature set where the features were based on the first 20 and 50 packets and a data set where all the asymmetric flows were removed, i.e. flows with packets in just one direction. The SVM method with RBF kernel has been in main focus in this study, in addition to Decision Tree and Random Forest in the parallel study.

A goal of this study was to see if it is possible to classify encrypted traffic, since encryption and other obfuscation techniques are used more and more. Unfortunately, within the timeframe of this study no such traffic data was at our disposal.

According to the achieved results I conclude that it is possible to classify the Internet traffic and achieve high performances by extracting characteristics based on the traffic flows. Overall accuracies above 92% were achieved. In addition recall and precision regarding Video was even higher in some cases.

Different sets of features have been evaluated and according to the results, presented in Chapter 6 and in Appendix A, payload sizes and statistical features based on payload sizes seem to be the features that have the most discriminative power. However, in order to decide which features are most suitable for the problem at hand more extensively testing regarding features need to be done. Furthermore, there may also be features that have not been considered during this study that would contribute better than the features that were used. How to select and compute the best features is a problem to resolve.

7.1. Future work

7.1.1. Encrypted traffic

As mentioned, no encrypted traffic was at hand during this study. However, it is of uttermost interest to see how a classifier can handle encrypted traffic.

7.1.2. Other machine learning methods

There are several more machine learning methods that could be used in order to classify Internet traffic which in a future study should be evaluated in order to find the method most suited for the problem. And in order to know which method is most suited the requirement of the classifiers from both a timewise perspective in addition to what errors are most important to reduce, for example it could be that the FPs regarding Video need to be as low as possible.

7.1.3. Alternative features

As mentioned earlier in the report, there could potentially be infinitely many features. And the problem of how to derive and decide which features are best is a problem to be resolved. Furthermore, in this study the features used have been packet based. The features could instead of packet based be flow based, as seen in some of the similar studies discussed in Chapter 3. For example features could be extracted and computed based on the first 120 seconds of a flow instead of the first 50 packets. Then feature such as number of packets in the flow, packets per second and the duration of the flow (note that some flows may be less than 120 seconds).

7.1.4. Different sampling and filtration

Through this study a payload packet was defined to be of at least 70 bytes in size. However, this may not have been the most optimal decision. Further testing to decide which size a payload packet should be defined as need to be done.

7.1.5. More Internet traffic classes

In this study the Internet traffic was divided to either belong to any of the two classes Video or Other. In Experiment 6 only 1 Internet service was defined as belonging to the Video class and the rest to Other class. The overall accuracy achieved was ~95% which indicates that the Video and possibly the Other class could be sub-divided further. For example, the video class could be divided into Live streaming and streaming and the Other class could be divided into gaming, HTTP, VoIP and Other. Another possibility is to not divide the Internet services at all and to use all the different services as class labels.

7.2. Concluding Remarks

In this study the ability to classify Internet traffic by extracting several flow characteristics has been explored. The method used was machine learning and the Support Vector Machine was the machine learning algorithm in focus. A total of 6 different experiments were conducted where in each experiment several tests were executed where different sets of features were evaluated. Results show that it is possible to achieve overall accuracies as high as ~90%.

Overall it was an interesting and educative project but difficult. In this project new techniques and methods were introduced. Machine learning is a big field and it had to be learned from scratch. To master machine learning, and be able to use it at its full potential, several years may be needed. In addition, the programming language Python and how to handle large amounts of data had to be learned.

References

[1] Internet Live Stats. [Online][Cited: 8 May 2016] http://www.internetlivestats.com/internet-users/#trend

[2] Thuy T.T. Nguyen; Grenville Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56-76, 2008.

[3] Tabatabaei Talieh Seyed; Karray Fakhri; Kamel Mohamed, "Early Internet Traffic Recognition based on Machine Learning Methods," in *Electrical & Computer Engineering (CCECE)*, Montreal, 2012, pp. 1-5.

[4] Alberto Dainotti; Antonio Pescapé; Carlo Sansone, "Early Classification of Network Traffic through Multi-classification," in *Traffic Monitoring and Analysis*, Jordi Domingo-Pascual, Yuval Shavitt, and Steve Uhlig, Eds. Vienna, Austria: Springer-Verlag Berlin Heidelberg, 2011, vol. 1, pp. 122-135.

[5] (2016) InternetAssignedNumbersAuthority. [Online][Cited: 11 2 2016] http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[6] (2016) Procera Networks. [Online][Cited: 11 2 2016] https://www.proceranetworks.com/index.php

[7] Henrik Johansson, "Video Flow Classification: Feature Based Classification Using the Tree Approach," Karlstad University, Waiting for approval.

[8] Peng Lizhi; Yang Bo; Chen Yuehui; Chen Zhenxiang, "Effectiveness of Statistical Features for Early Stage Internet Traffic Identification," *International Journal of Parallel Programming* , vol. 44, no. 1, pp. 181-197, 2015.

[9] Jeankyung Kim; Jinsoo Hwang; Kichang Kim, "High-Performance Internet Traffic Classification Using a Markov Model and Kullback-Leibler Divergence," *Mobile Information Systems*, 2015.

[10] Jayeeta Datta; Neha Kataria; Neminath Hubballi, "Network Traffic Classification in Encrypted Enviroment: A Case Study of Google Hangout," Department of Computer Science and Engineering, Indian Institute of Technology Indore, 2015.

[11] (2016) Swedish National Infrastructure for Computing. [Online][Cited: 11 2 2016] http://www.snic.se/

[12] (2016) Jupyter - IPython Notebook. [Online][Cited: 6 May 2016] http://jupyter.org/

[13] (2016) Pandas. [Online][Cited: 7 May 2016] http://pandas.pydata.org/

[14] (2013) Numpy. [Online][Cited: 7 May 2016] http://www.numpy.org/

[15] (2015) Scikit-learn. [Online][Cited: 7 May 2016] http://scikit-learn.org/stable/#

[16] Trevor Hastie; Robert Tibshirani; Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed.: Springer, 2009.

[17] Luis Pedro Coelho; Willi Richert, *Building Machine Learning Systems with Python*.: Packt Publishing Ltd., 2015.

[18] Phil Simon, *Too Big to Ignore: The Business Case for Big Data*.: Wiley, 2013.

[19] Tom Mitchell, *Machine Learning*., 1997.

[20] Ron Kohavi. (1998) Machine Learning Glossary. [Online][Cited: 11 2 2016] http://robotics.stanford.edu/~ronnyk/glossary.html

[21] Jason Brownlee. (2014) machinelearningmastery. [Online][Cited: 6 May 2016] http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

[22] Scott Fortmann-Roe, Understanding the Bias-Variance Tradeoff, 2012, An essay by Scott Fortmann-Roe, http://scott.fortmann-roe.com/.

[23] Brendan Duncan. FlipTop. [Online][Cited: 17 Feb 2016] https://blog.fliptop.com/blog/2015/03/02/bias-variance-and-overfitting-machine-learning-overview/

[24] (2014) Scikit-learn - Linear SVM. [Online][Cited: 7 May 2016] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[25] (2015) Wikipedia, Multiclass Classification. [Online][Cited: 6 May 2016] https://en.wikipedia.org/wiki/Multiclass_classification

[26] Chrisopher J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[27] Alexandre Kowalczyk. (2014) SVM Tutorial (math behind SVM). [Online][Cited: 7 May 2016] http://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/

[28] Eric Kim. (2013) SVM-Kernel Trick. [Online][Cited: 6 May 2016] http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

[29] S. B. Kotsiantis, "Supervised Machine Learning: A review of Classification Techniques," in *Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, Peloponnese, 2007, pp. 3-24.

[30] Thomas G. Dietterich, "Ensemble Methods in Machine Learning," Oregon State University, Oregon,.

[31] Dan Benyamin. (2012, Oct.) CitizenNet. [Online][Cited: 6 May 2016] https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

[32] Marting Sewell, "Ensemble Learning," Department of Computer Science, Univeristy College London, 2007.

[33] Deepanshu Bhalla. Listen Data. [Online][Cited: 6 May 2016] http://www.listendata.com/2014/11/random-forest-with-r.html

[34] Jason Brownlee. (2013) MachineLearningMastery. [Online][Cited: 6 May 2016] http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

[35] (2016) Instance Based Learning - Wikipedia. [Online][Cited: 7 May 2016] https://en.wikipedia.org/wiki/Instance-based_learning

[36] Saravanan Thirumuruganathan. (2010) wordpress - k nearest neghbors. [Online][Cited: 7 May 2016] https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

[37] Will Kurt. (2015, Feb.) Bayes Theorem. [Online][Cited: 7 May 2016] https://www.countbayesie.com/blog/2015/2/18/bayes-theorem-with-lego

[38] Richard O. Duda; Peter E. Hart; David G. Stork, *Pattern Classification*, 2nd ed.: Wiley, 2000.

[39] Gu Chengjie; Zhang Shunyi; Huang He, "Online Internet Traffic Classification Based on Proximal SVM," *Journal of Computational Information System 7*, vol. 6, pp. 2078-2086, 2011.

[40] Li Jun; Zhang Shunyi; Lu Yanqing; Zhang Zailong, "Internet Traffic Classification Using Machine Learning," in *Communications and Networking*, Shanghai, 2007, pp. 239-243.

[41] Jaiswal Rupesh Chandrakand; Lokhande Shashikant D., "Machine Learning Based Internet Traffic Recognition with Statistical Approach," Department of Electronics & Telecommunication, Institute of Computer Technology , Pune, Inida, 978-1-4799-2274-1/2325-940X, 2013.

[42] Zhu Li; Ruixi Yuan; Xiaohong Guan, "Accurate Classification of the Internet Traffic Based on the SVM Method," in *2007 IEEE International Conference on Communications*, Glasgow, 2007, pp. 1373-1378.

[43] Kuldeep Singh; Sunil Agrawal, "Comperative analasys of five machine learning algorithms for IP traffic classification," in *Emerging Trends in Networks and Computer Communications (ETNCC)*, Udaipur, 2011, pp. 33-38.

[44] Tom Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers," Intelligent Enterprise Technologies Laboratory, HP Laboratories, Palo Alto, 2003.

[45] Thomas G. Tape. Interpreting Diagnostic Tests. [Online][Cited: 7 May 2016] http://gim.unmc.edu/dxtests/Default.htm

[46] (2014) Scikit-learn - SVM. [Online][Cited: 7 May 2016] http://scikit-learn.org/stable/modules/svm.html

[47] (2015) Platt Scaling - Wikipedia. https://en.wikipedia.org/wiki/Platt_scaling

[48] John C. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Advances in Large Margin Classifiers*, pp. 61-74, 1999.

[49] Ting-Fan Wu; Chih-Jen Lin; Ruby C. Weng, "Probability Estimates for Multi-class Classification By Pairwise Coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975-1005, 2004.

[50] Payam Refaeilzadeh; Lei Tang; Huan Liu, "Cross-Validation," Arizona State University, 2008.

[51] (2013) Cross-validation: right and wrong. [Online][Cited: 18 May 2016] https://www.youtube.com/watch?v=S06JpVoNaA0

[52] Warren S. Sarle. (2002) NeuralNets - Faqs. [Online][Cited: 7 May 2016] http://www.faqs.org/faqs/ai-faq/neural-nets/

[53] (2014) Scikit-learn - Preprocessing. [Online][Cited: 7 May 2016] http://scikit-learn.org/stable/modules/preprocessing.html

[54] Chih-Wei Hsu; Chih-Chung Chang; chih-Jen Lin, "A Practical Guide to Support Vector Classification," Department of Computer Science, National Taiwan University, Taiwan, 2010.

[55] (2014) Sklearn - Feature Selection. [Online][Cited: 16 May 2016] http://scikit-learn.org/stable/modules/feature_selection.html

[56] Chih-Chung Chang; Chih-Jen Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, 2011, Article No. 27.

[57] Rong-En Fan; Kai-Wei Chang; Cho-Jui Hsieh;Xiang-Rui Wang; Chih-Jen Lin, "LIBLINEAR: A Library for Large Linear Classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871-1874, 2015.

[58] (2014) Scikit-learn - SVM RBF Parameters. [Online][Cited: 7 May 2016] http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[59] (2014) scikit-learn. [Online][Cited: 6 May 2016] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Appendix A

In Appendix A the results for the rest of the experiments conducted during this study are discussed. Each sub-section shows the results for a particular experiment and each experiment includes which initial feature set that were used together with which data set and a summarization of the filtered data set. Including specifics about the filtration, which ML algorithm that were used and which feature selection method that was used. Each experiment is comprised of several different tests and each test shows which features that were used and also the performance of the model that was chosen.

A.1. Experiment 2 - Linear SVC with packet feature set: 20 first packets

Algorithm: Linear SVC

Initial feature set: Packet feature set

The features were computed based on the first 20 packets. Features were computed at packet indices 6, 8, 10, 12, 14, 16, 18 and 20.

Sampled data set: *10k_10k*

Filtering specifics: The payload size was defined as >70 bytes. Only traffic flows with at least 10 packets were considered.

Filtered data set:

```
Number of Unique Flows   :  16670
Number of Unique Services:  129

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |      7617 | 45.69
HTTP                                    |      2260 | 13.56
BitTorrent transfer                     |       880 | 5.28
uTP                                     |       769 | 4.61
Youku                                   |       700 | 4.20
MiBox                                   |       678 | 4.07
DNS                                     |       422 | 2.53
BitTorrent KRPC                         |       408 | 2.45
RTMP                                    |       299 | 1.79
PPStream                                |       285 | 1.71

Freq Video

SERVICE   |Count      | Share (%)
_____________________________________
Video     |     10359 | 62.14
Other     |      6311 | 37.86
```

A.1.1. Test 1 - All features

In Test 1 all of the 174 features in the initial feature set were used. This test can be seen as the benchmark test in order to see how well one can expect the classifier to perform. As can be seen in Figure 46 the value of the parameter C does not seem to have a great impact on the performance and a value of 1 for C was used. The performance for the corresponding classifier can be seen in the confusion matrix. An overall accuracy of 84.59% was achieved. However, the recall and precision regarding Video were a bit higher, 88.29% and 87.12% respectively.

Feature selection: No feature selection.

Features: All features in the initial feature set, a total of 174 features.

Table 13 Exp. 2 Test 1 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	4941	1354	6295	78.49%	80.26%	
	Video	1215	9160	10375	88.29%	87.12%	
	Total	6156	10514	16670			84.59%

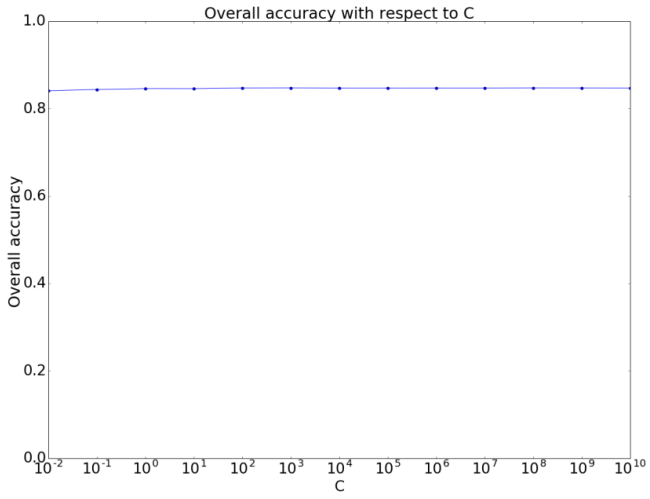

Figure 46 Exp. 2 Test 1. Overall accuracy while varying the parameter C.

A.1.2. Test 2 – Payload size based features

In Test 2 a total of 13 features were used. The features used were based on the ten first payload sizes and these features were tested because similar studies has suggested that individual payload sizes and statistical features based on payload sizes are features that should not be ignored. The parameter C was set to 0.01 by inspection of Figure 47. The performance for the corresponding classifier can be seen in the confusion matrix. An overall accuracy of 75.78% was achieved, which is deemed low, furthermore the precision regarding Other is as low as 65.41% and these low scores are an indication that the data is nonlinear separable.

Feature selection: No feature selection.

Features: 13 features in total comprised of the ten first individual payload sizes in addition to three statistical features based on the payload sizes.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylMean, PaylVar, PaylStd}

Table 14 Exp. 2 Test 2 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	4792	1503	6295	76.12%	65.41%	
	Video	2534	7841	10375	75.58%	83.91%	
	Total	7326	9344	16670			75.78%

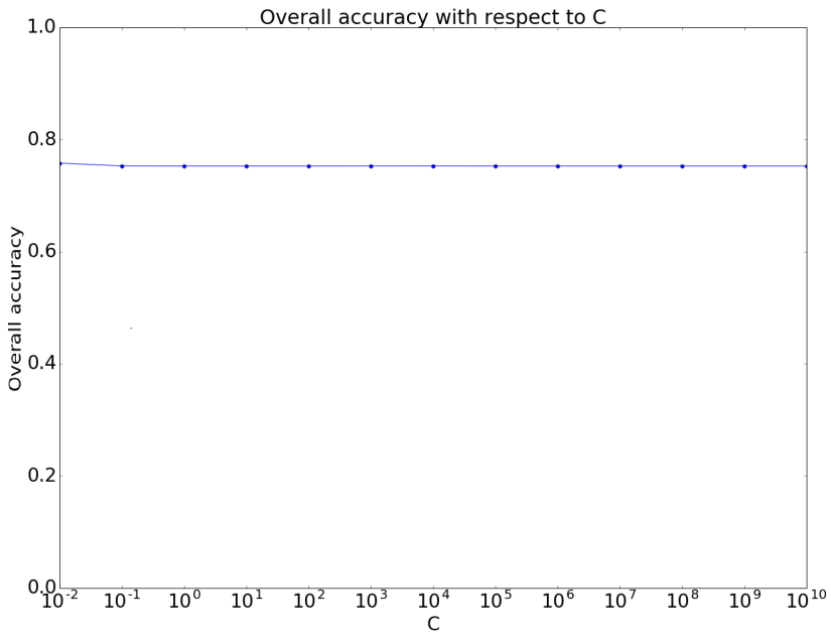

Figure 47 Exp. 2 Test 2 Overall accuracy while varying the parameter C.

A.2. Experiment 3 - SVC with RBF kernel using payload feature set : 10 first payload packets

Algorithm: SVC with RBF kernel.

Initial feature set: Payload feature set.

The features were computed based on the first ten payload packets, except for the direction feature which were computed based on the first 8 packets. The initial feature set consisted of 291 features total.

Sampled data set: *10k_10k.*

Filtering specifics: The payload size was defined as >70 bytes. Only traffic flows with at least 10 payload packets were considered.

Filtered data set:

```
Total Number of Packets  :  181520
Number of Unique Flows   :  9076
Number of Unique Services:  87

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |      6237 | 68.72
MiBox                                   |       661 | 7.28
HTTP                                    |       596 | 6.57
PPStream                                |       183 | 2.02
BitTorrent KRPC                         |       181 | 1.99
Flash video over HTTP                   |       163 | 1.80
Teredo                                  |       132 | 1.45
Youku                                   |        92 | 1.01
DNS                                     |        80 | 0.88
Kodi                                    |        78 | 0.86

Freq Video

SERVICE   |Count      | Share (%)
_____________________________________
Video     |      7585 | 83.57
Other     |      1491 | 16.43
```

A.2.1. Test 1 - Feature Selection by Variance

In Test 1 the features used were selected by variance feature selection and all of the features, except for the larger part of the direction features, were selected. The combination of values for the parameters C and gamma was searched for using grid search and the result can be seen in the heat maps in Figure 48 and in Figure 49. According to these results the best values for both C and gamma were 10. In Table 15 the confusion matrix which shows the results for the corresponding classifier can be seen. An overall accuracy of 91.97% was achieved and the recall and precision regarding Video is even higher. Remarkably is the precision which reached as high as 96.61%.

Feature selection: The variance feature selection algorithm was used and after selection 35 features were selected. The features removed were all but two of the direction features.

Table 15 Exp. 3 Test 1 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	1242	249	1491	83.30%	72.13%	
	Video	480	7105	7585	93.67%	96.61%	
	Total	1722	7354	9076			91.97%

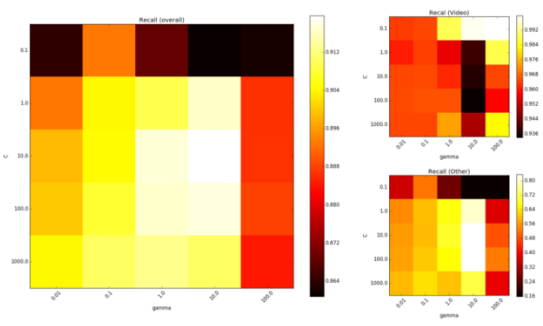

Figure 48 Exp. 3 Test 1 recall while varying C and gamma.

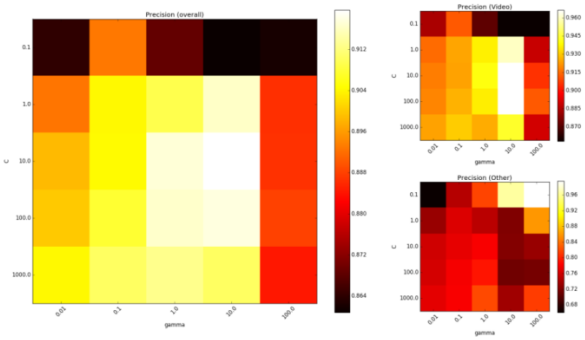

Figure 49 Exp. 3 Test 1 precision while varying C and gamma.

A.3. Experiment 4 – SVC with RBF kernel using packet feature set: 50 first packets

Algorithm: SVC with RBF kernel

Initial feature set: Packet feature set

The features were computed based on the first 50 packets. Features were computed at packet indices 30, 32, 34, 36, 38, 40, 42, 44, 46, 48 and 50. Furthermore the inter arrival time features were excluded in this experiment.

Sampled data set: *100k_100k*

Filtering specifics: The payload size was defined as >70 bytes. Only traffic flows with at least 50 packets were considered.

Further sampling: Because of how the SVC scales with more samples, recall Figure 19, the data set was further samples to include around 9000 Video flows and 9000 Other flows.

Filtered data set:

```
Number of Unique Flows   :  18080
Number of Unique Services:  142

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |      7403 | 40.95
HTTP                                    |      4506 | 24.92
uTP                                     |       986 | 5.45
BitTorrent KRPC                         |       735 | 4.07
MiBox                                   |       695 | 3.84
Test pattern                            |       424 | 2.35
BitTorrent transfer                     |       357 | 1.97
Flash video over HTTP                   |       285 | 1.58
Teredo                                  |       206 | 1.14
PPStream                                |       192 | 1.06

Freq Video

SERVICE   |Count      | Share (%)
_____________________________
Video     |      9000 | 49.78
Other     |      9080 | 50.22
```

A.3.1. Test 1 – All features

In Test 1 all of the features in the initial feature set were used. The values for C and gamma were selected to be 100 and 0.1 respectively by inspection of the heat maps seen in Figure 51 and Figure 52. The performance achieved for the classifier in Test 1 can be seen as a benchmark score. The evaluation result for the classifier is summarized in the confusion matrix in Table 16 and in the ROC curve in Figure 50. An overall accuracy of 91.90% was achieved which is just a little higher than in the benchmark test in *Experiment 1 - SVC with RBF kernel using packet feature set: 20 first packets*, however, the recall and precision scores between the two classes are more even. With that said, it seems that the overall performance when considering a larger range of packets is not improved significantly.

Features: 172 features total.

Table 16 Exp. 4 Test 1 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	8408	672	9080	92.60%	91.39%	
	Video	792	8208	9000	91.20%	92.43%	
	Total	9200	8880	18080			91.90%

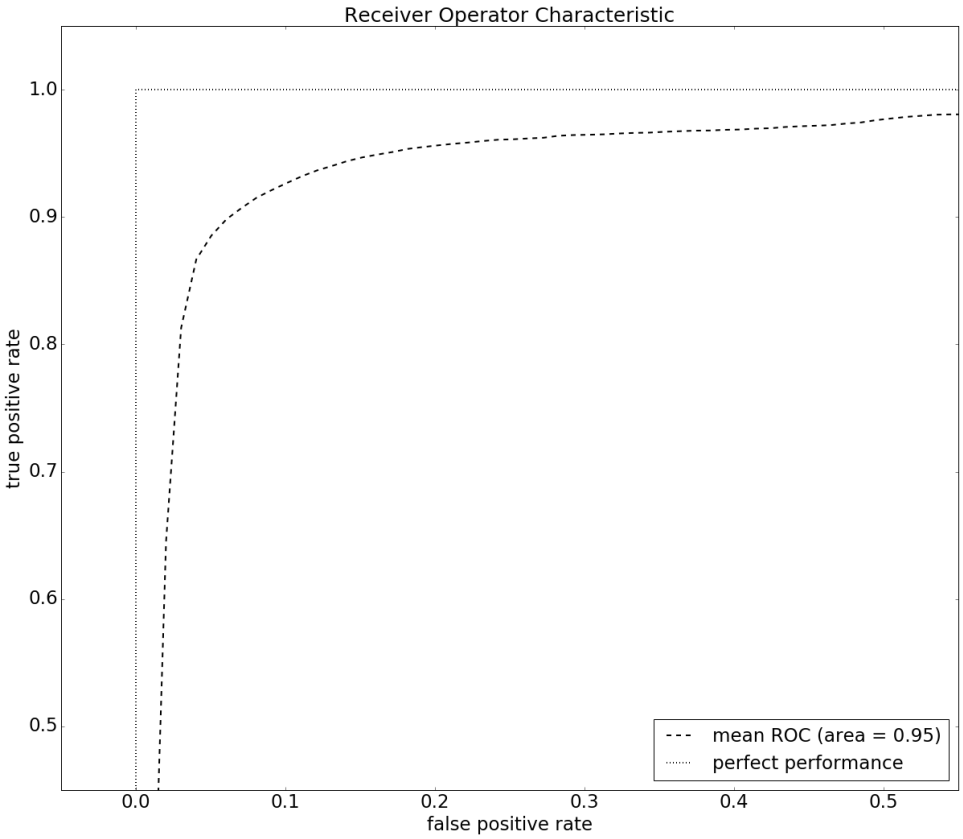

Figure 50 Exp.4 Test 1 ROC curve.

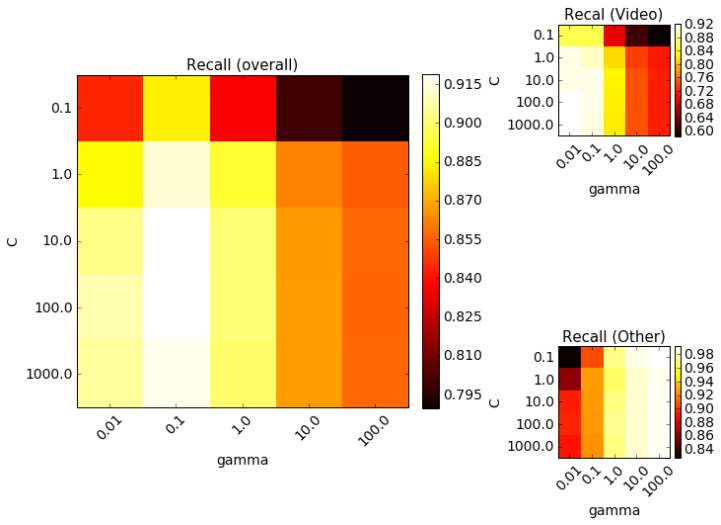

Figure 51 Exp. 4 Test 1 recall while varying C and gamma.

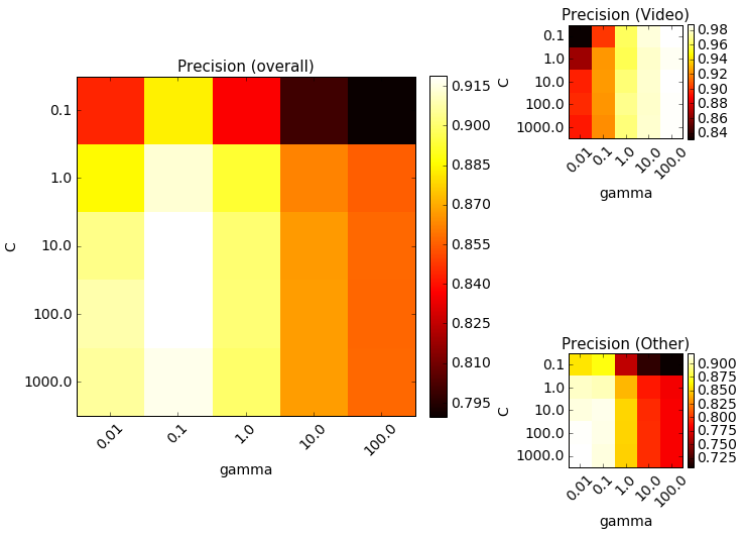

Figure 52 Exp. 4 Test 1 precision while varying C and gamma.

A.3.2. Test 2 - Features based on the first 50 packets

In Test 2 all the features computed based on the first 50 packets were used constituting a total of 13 features. These 13 features were chosen in order to see how well the classifier could perform when considering features based on the first 50 packets. By inspection of the heat maps in Figure 54 and Figure 55 the best values for C and gamma were chosen as 1 and 10 respectively. The performance for the classifier can be seen in the confusion matrix in Table 17 and in the ROC curve in Figure 53. An overall accuracy of 88.57% was achieved and the recall and precision for both classes are in the same magnitudes.

Feature selection:

Features: 13 features total

Feature set = {BytesDwUp-50, BytesDw-50, BytesDwPayl-50, NrDownPkts-50, NrDownPaylPkts-50, PosSumDwnPkts-50, PosSumDwnPaylPkts-50, PosSumDwnPktsEven-50, PosSumDwnPaylPktsEven-50, StdDevDwnPkts-50, StdDevDwnPaylPkts-50, DwnPaylRun-50, DwnPaylRun200-50}

Table 17 Exp. 4 Test 2 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	8070	1010	9080	88.88%	88.42%	
	Video	1057	7943	9000	88.26%	88.72%	
	Total	9127	8953	18080			88.57%

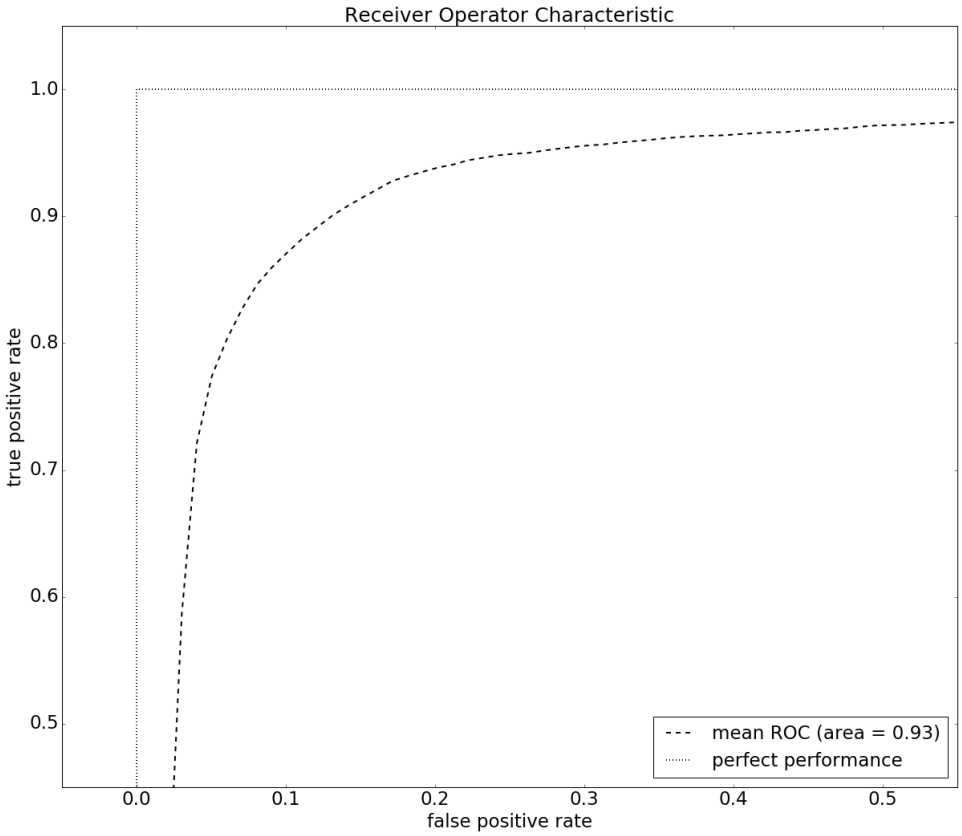

Figure 53 Exp. 4 Test 2 ROC curve.

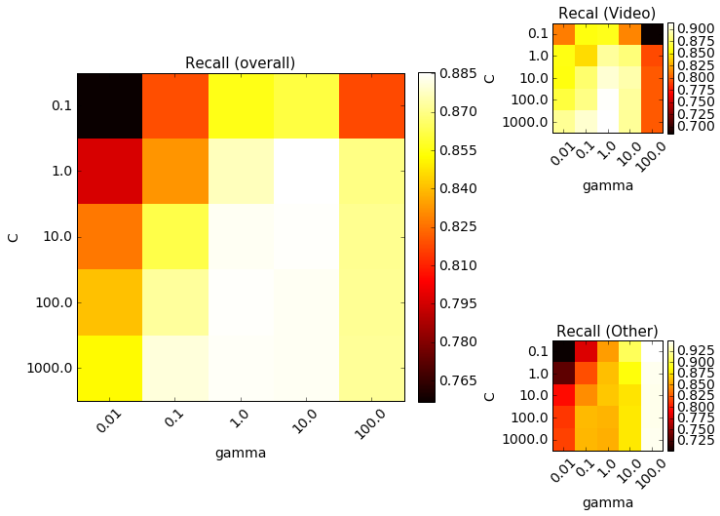

Figure 54 Exp. 4 Test 2 recall while varying C and gamma.

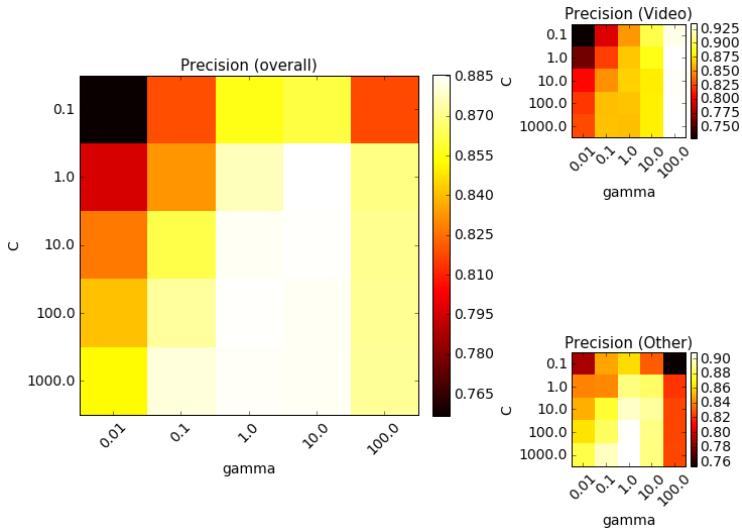

Figure 55 Exp. 4 Test 2 precision while varying C and gamma.

A.3.3. Test 3 – Single feature

In Test 3 the only feature used was the one that the Decision Tree algorithm ranked highest. The best values for C and gamma was both decided by inspection of the heat maps in Figure 56 and Figure 57 to be 100. The performance for the classifier is shown in the confusion matrix in Table 18 and as can be seen an overall accuracy of 79.70% was achieved which is not that impressive.

Feature selection: Most important feature according to the Decision Tree classifier.

Features: 1 feature total.

Feature set = {DwnPaylRun200-50}

Table 18 Exp. 4 Test 3 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	7710	1370	9080	84.91%	77.02%	
	Video	2300	6700	9000	74.44%	83.02%	
	Total	10010	8070	18080			79.70%

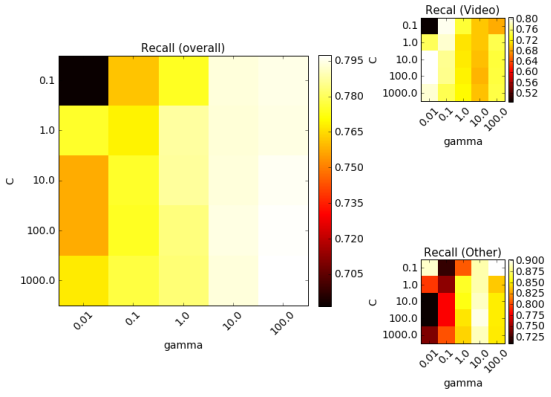

Figure 56 Exp. 4 Test 3 recall while varying C and gamma.

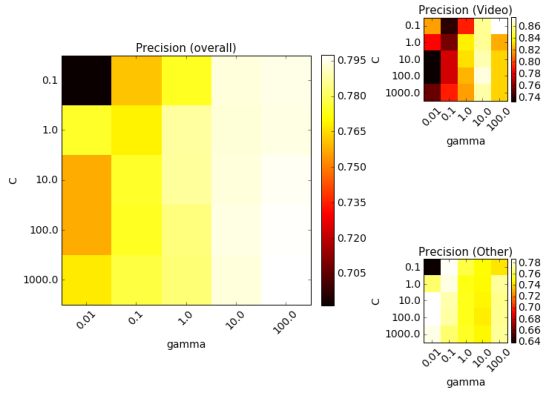

Figure 57 Exp. 4 Test 3 precision while varying C and gamma.

A.3.4. Test 4 – The downward payload run features

In Test 4 all of the downward payload run features were used comprising a total of 22 features. C and gamma were chosen as 1 and 10 respectively by inspection of the heat maps seen in Figure 59 and Figure 60. The performance for the corresponding classifier can be seen in the confusion matrix in Table 19 in addition to the ROC curve in Figure 58. Considering that 22 features were used the overall accuracy of 82.53% is deemed bad.

Feature selection: All of the downward payload run features.

Features: 22 features in total

Feature set = {DwnPaylRun-30, DwnPaylRun-32, DwnPaylRun-34, DwnPaylRun-36, DwnPaylRun-38, DwnPaylRun-40, DwnPaylRun-42, DwnPaylRun-44, DwnPaylRun-46, DwnPaylRun-48, DwnPaylRun-50, DwnPaylRun200-30, DwnPaylRun200-32, DwnPaylRun200-34, DwnPaylRun200-36, DwnPaylRun200-38, DwnPaylRun200-40, DwnPaylRun200-42, DwnPaylRun200-44, DwnPaylRun200-46, DwnPaylRun200-48, DwnPaylRun200-50}

Table 19 Exp. 4 Test 4 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	7922	1158	9080	87.25%	79.83%	
	Video	2001	6999	9000	77.77%	85.80%	
	Total	9923	8157	18080			82.53%

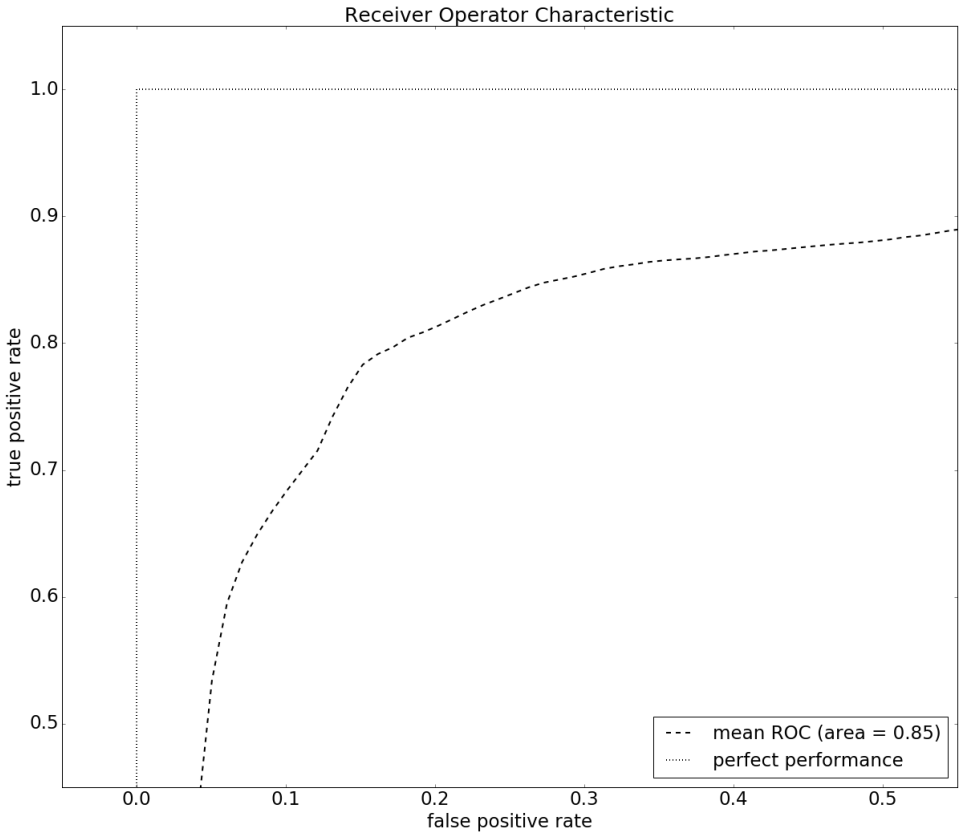

Figure 58 Exp. 4 Test 4 ROC curve.

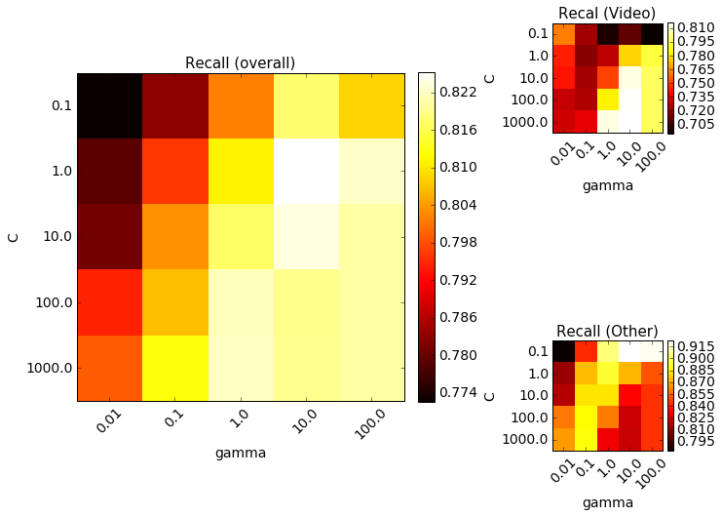

Figure 59 Exp. 4 Test 4 recall while varying C and gamma.

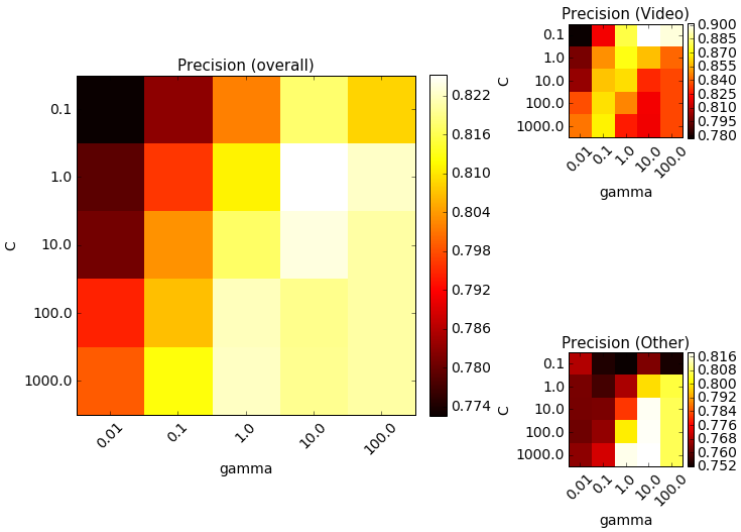

Figure 60 Exp. 4 Test 4 precision while varying C and gamma.

A.3.5. Test 5 – Payload size based features

In Test 5 the features used were the first 25 payload sizes and three statistical features based on these payload sizes comprising a total of 28 features. The best values for both C and gamma were chosen as 100 by inspection of the heat maps in Figure 62 and Figure 63. In the confusion matrix and the ROC curve in Figure 61 the performance for the classifier can be seen. An overall accuracy of 91.43% was achieved, which is just a little bit under the benchmark accuracy. The recall for Other is as high as 95.90% and the precision regarding Video is almost as high. However, on the downside the recall for Video is only 86.92% which is a little low. In addition the ROC curve is steep as well but the AUC score is a little lower than the benchmark score.

Feature selection: Individual payload sizes for 25 payload packets in addition to the three statistical features average, variance and standard deviation based on the individual payload sizes.

Features: 28 features total

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylSize-11, PaylSize-12, PaylSize-13, PaylSize-14, PaylSize-15, PaylSize-16, PaylSize-17, PaylSize-18, PaylSize-19, PaylSize-20, PaylSize-21, PaylSize-22, PaylSize-23, PaylSize-24, PaylSize-25, PaylMean, PaylVar, PaylStd}

Table 20 Exp. 4 Test 5 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	8708	372	9080	95.90%	88.09%	
	Video	1177	7823	9000	86.92%	95.46%	
	Total	9885	8195	18080			91.43%

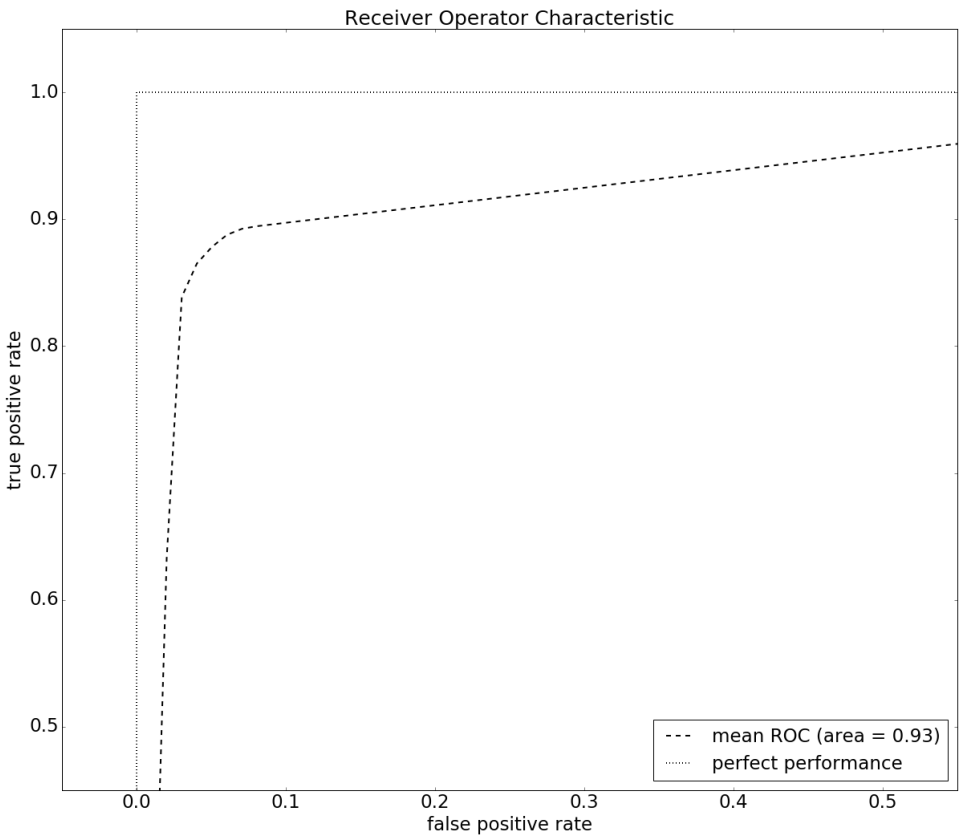

Figure 61 Exp. 4 Test 5 ROC curve.

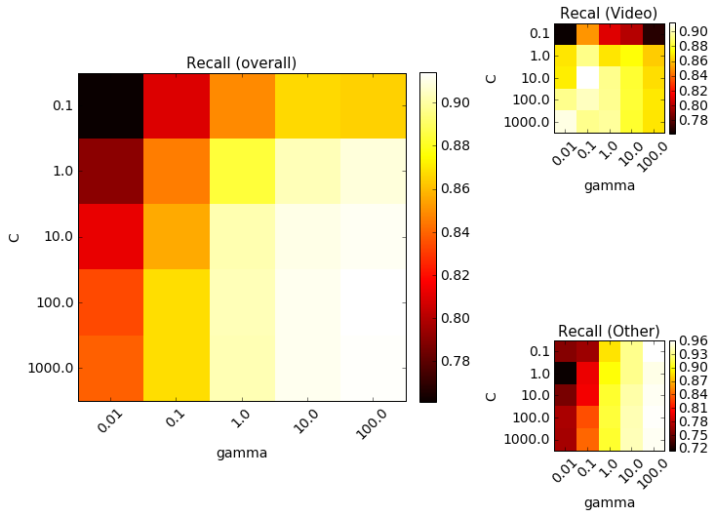

Figure 62 Exp. 4 Test 5 recall while varying C and gamma.

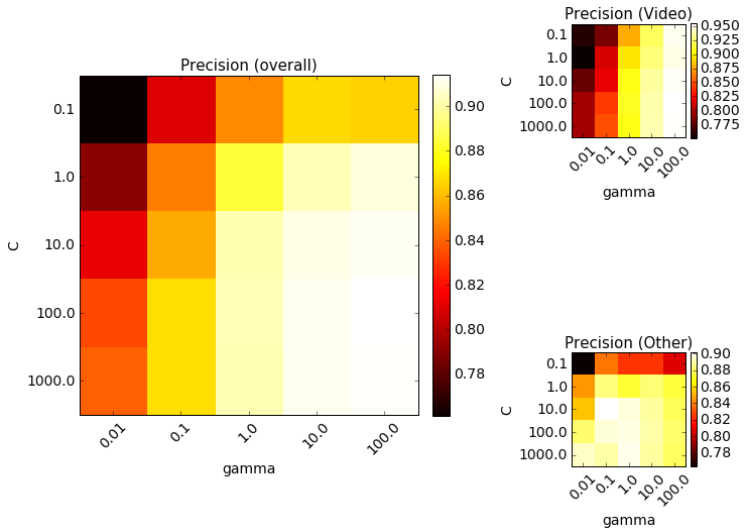

Figure 63 Exp. 4 Test 5 precision while varying C and gamma.

A.3.6. Test 6 – Payload based features

In Test 6 a total of 8 features were used; three statistical features based on the first 25 payload packets together with four other features that are based on the payload packets. By inspection of the heat maps seen in Figure 65 and Figure 66 C and gamma were chosen as 10 and 100 respectively. The performance for the corresponding classifier can be seen in the confusion matrix and in the ROC curve in Figure 64. Considering that only 7 features were used an overall accuracy of 90.60% is good if compared to Test 5 and the benchmark test, i.e. Test 1. The ROC curve is steep and the AUC, 0.95, score is also good. However, the recall for video and the precision for Other are a little bit low.

Feature selection: The three statistical features average, variance and standard deviation based on the payload packets in addition to four payload based features.

Features: 7 features total

Feature set = {PaylMean, PaylVar, PaylStd, DwnPaylRun200-50, BytesDwPayl-50, PosSumDwnPaylPkts-50, NrDownPaylPkts-50}

Table 21 Exp. 4 Test 6 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	8682	398	9080	95.62%	86.96%	
	Video	1302	7698	9000	85.53%	95.08%	
	Total	9984	8096	18080			90.60%

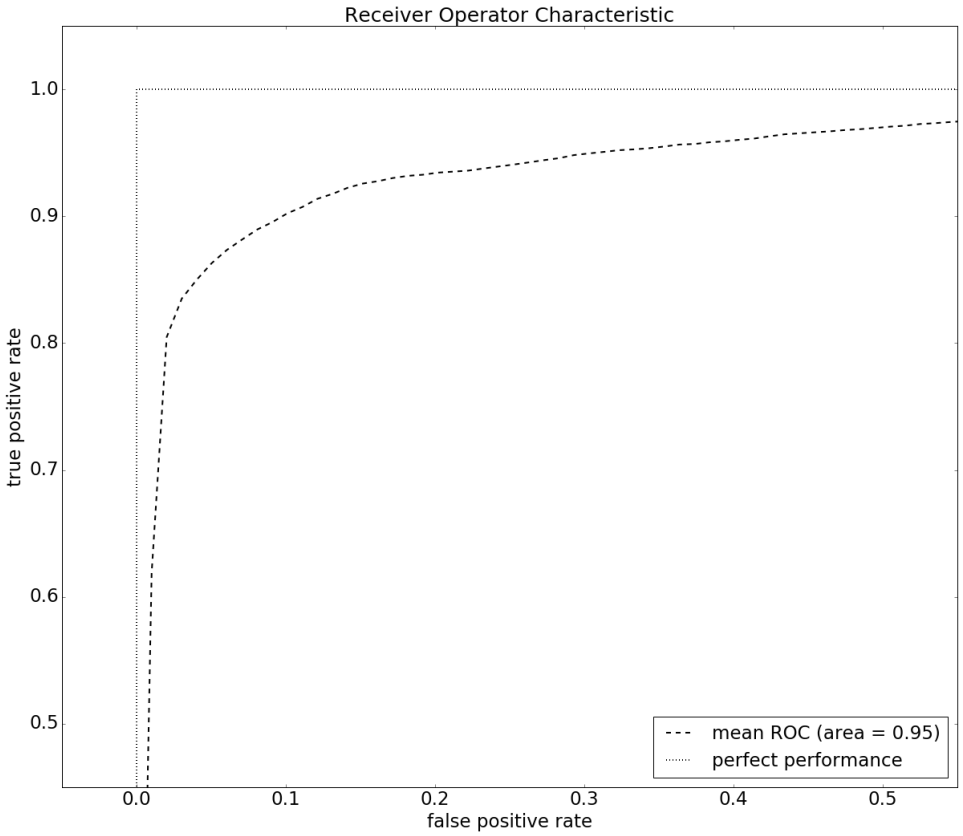

Figure 64 Exp. 4 Test 6 ROC curve.

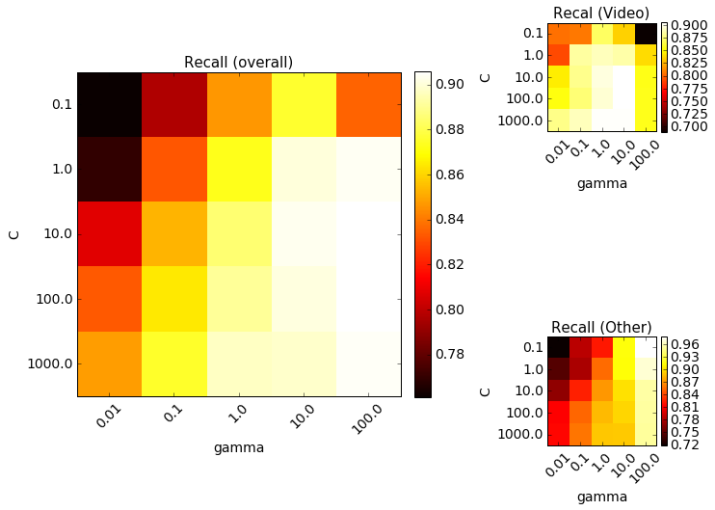

Figure 65 Exp. 4 Test 6 recall while varying C and gamma.

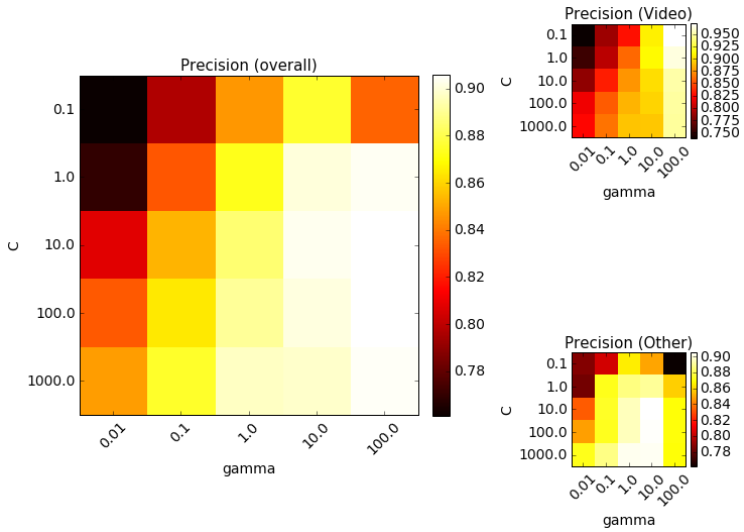

Figure 66 Exp. 4 Test 6 precision while varying C and gamma.

A.3.7. Test 7 – The position sum features

In Test 7 all of the position sum features were used, a total of 44 features. By inspection of the heat maps in Figure 68 and Figure 69 the values for C and gamma were chosen as 100 and 10 respectively. The performance for the corresponding classifier is seen in the confusion matrix and in the ROC curve in Figure 67. Considering the overall accuracy of 83.02% the position sum features seems to be insufficient.

Feature selection: All the position sum features.

Features: 44 features in total.

Feature set = {PosSumDwnPkts-30, PosSumDwnPkts-32, PosSumDwnPkts-34, PosSumDwnPkts-36, PosSumDwnPkts-38, PosSumDwnPkts-40, PosSumDwnPkts-42, PosSumDwnPkts-44, PosSumDwnPkts-46, PosSumDwnPkts-48, PosSumDwnPkts-50, PosSumDwnPaylPkts-30, PosSumDwnPaylPkts-32, PosSumDwnPaylPkts-34, PosSumDwnPaylPkts-36, PosSumDwnPaylPkts-38, PosSumDwnPaylPkts-40, PosSumDwnPaylPkts-42, PosSumDwnPaylPkts-44, PosSumDwnPaylPkts-46, PosSumDwnPaylPkts-48, PosSumDwnPaylPkts-50, PosSumDwnPktsEven-30, PosSumDwnPktsEven-32, PosSumDwnPktsEven-34, PosSumDwnPktsEven-36, PosSumDwnPktsEven-38, PosSumDwnPktsEven-40, PosSumDwnPktsEven-42, PosSumDwnPktsEven-44, PosSumDwnPktsEven-46, PosSumDwnPktsEven-48, PosSumDwnPktsEven-50, PosSumDwnPaylPktsEven-30, PosSumDwnPaylPktsEven-32, PosSumDwnPaylPktsEven-34, PosSumDwnPaylPktsEven-36, PosSumDwnPaylPktsEven-38, PosSumDwnPaylPktsEven-40, PosSumDwnPaylPktsEven-42, PosSumDwnPaylPktsEven-44, PosSumDwnPaylPktsEven-46, PosSumDwnPaylPktsEven-48, PosSumDwnPaylPktsEven-50}

Table 22 Exp. 4 Test 7 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	7639	1441	9080	84.13%	82.42%	
	Video	1629	7371	9000	81.90%	83.65%	
	Total	9268	8812	18080			83.02%

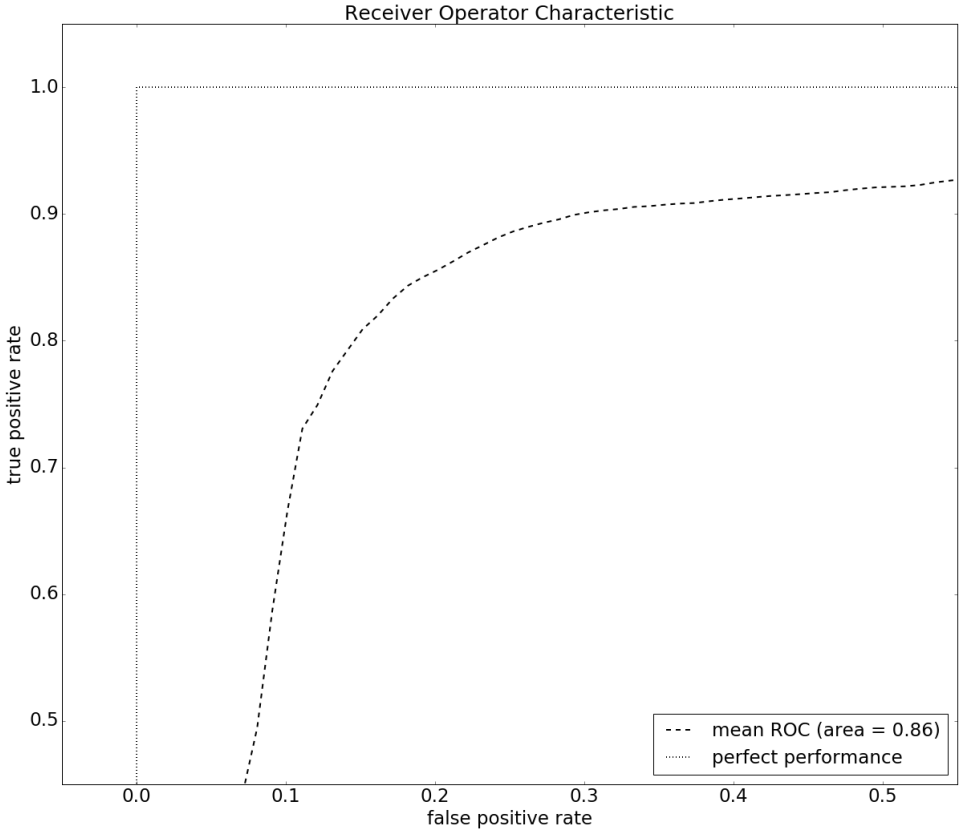

Figure 67 Exp. 4 Test 7 ROC curve.

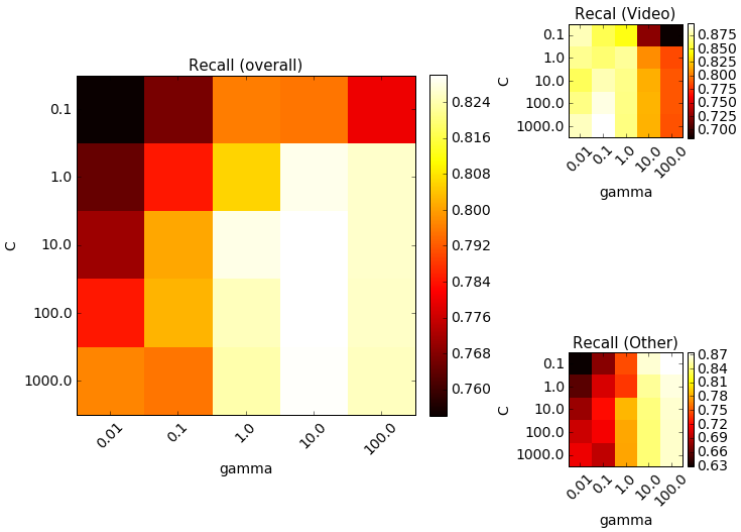

Figure 68 Exp. 4 Test 7 recall while varying C and gamma.

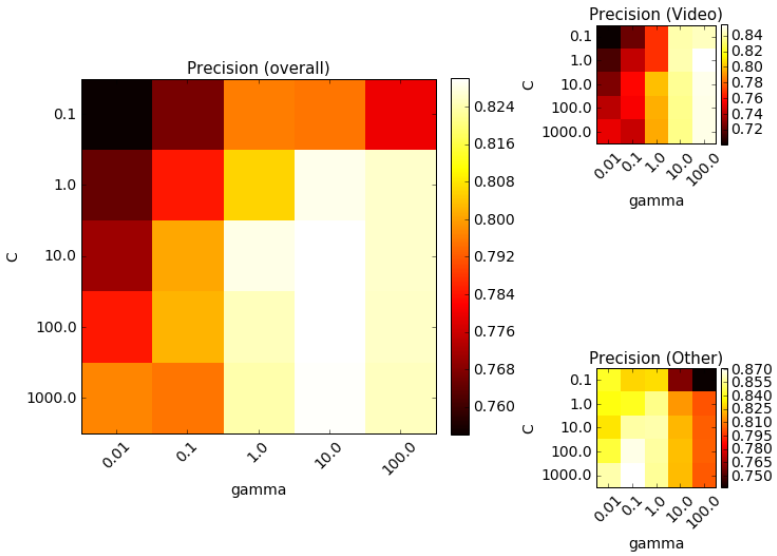

Figure 69 Exp. 4 Test 7 precision while varying C and gamma.

A.4. Experiment 5 – Asymmetric flows removed SVC with RBF kernel using packet feature set: 20 first packets

Algorithm: SVC with RBF kernel

Initial feature set: Packet feature set

The features were computed based on the first 20 packets. Features were computed at packet indices 6, 8, 10, 12, 14, 16, 18, 20.

Sampled data set: *100k_100k*

Filtering specifics: The payload size was defined as >70 bytes. Only traffic flows with at least 10 packets were considered.

Further sampling: Asymmetric flows were removed. Because of how the SVC scales with more samples, recall Figure 19, the data set was further samples to include around 10000 Video flows and 10000 Other flows.

Filtered data set:

```
Number of Unique Flows   :  20000
Number of Unique Services:  129

Freq of Top 10 Internet services:

SERVICE                                 |Count      | Share (%)
_____________________________________________________________
HTTP media stream                       |      6441 | 32.20
HTTP                                    |      3523 | 17.61
BitTorrent transfer                     |      1682 | 8.41
uTP                                     |      1147 | 5.73
Youku                                   |      1028 | 5.14
BitTorrent KRPC                         |       670 | 3.35
DNS                                     |       619 | 3.09
RTMP                                    |       524 | 2.62
BitTorrent encrypted transfer           |       505 | 2.53
Kodi                                    |       416 | 2.08

Freq Video

SERVICE   |Count      | Share (%)
_____________________________________
Video     |     10000 | 50.00
Other     |     10000 | 50.00
```

A.4.1. Test 1 – All features

In Test 1 all of the features in the initial feature set were used. Test 1 can be seen as the benchmark test. The parameters C and gamma were chosen as 10 and 0.1 respectively by inspection of the heat maps in Figure 71 and Figure 72. The performance for the corresponding classifier can be seen in the confusion matrix and in the ROC curve in Figure 70. In comparison to the benchmark test in *Experiment 1 - SVC with RBF kernel using packet feature set: 20 first packets* a higher overall accuracy was achieved, an accuracy of 92.56%, while the recall and precision regarding Video is worse. However, the difference in performances probably depends on the differences in the data sets that were used.

Features: 134 features total.

Feature selection: All features.

Table 23 Exp. 5 Test 1 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	9366	634	10000	93.66%	91.64%	
	Video	854	9146	10000	91.46%	93.52%	
	Total	10220	9780	20000			92.56%

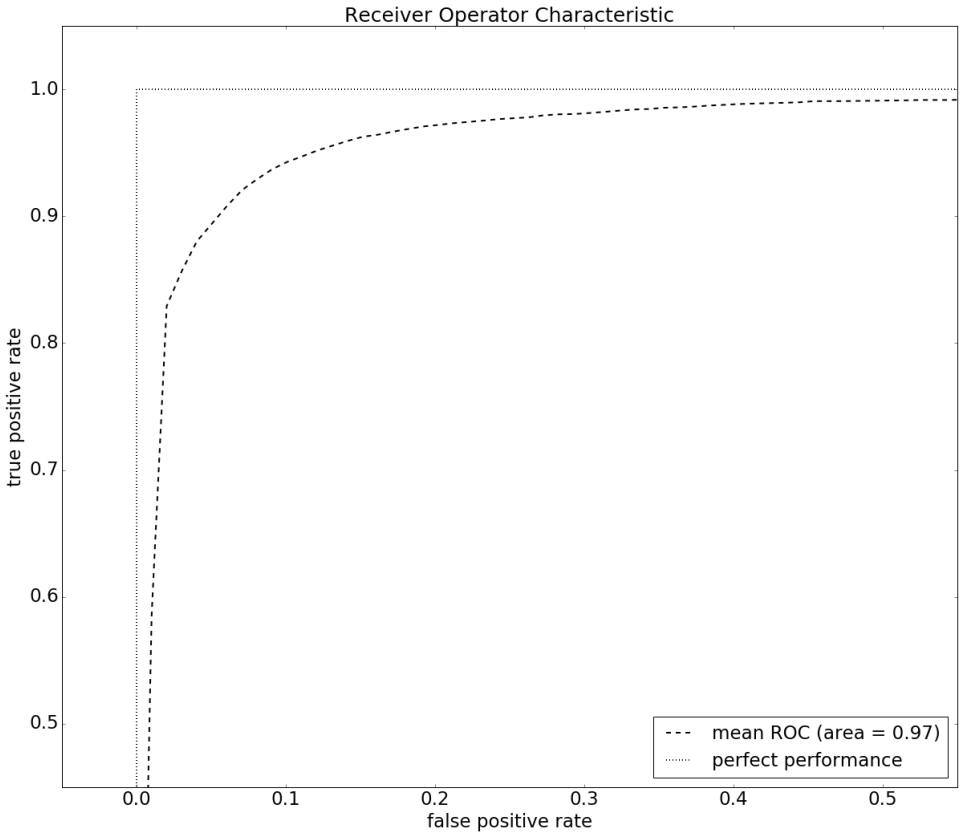

Figure 70 Exp. 5 Test 1 ROC curve.

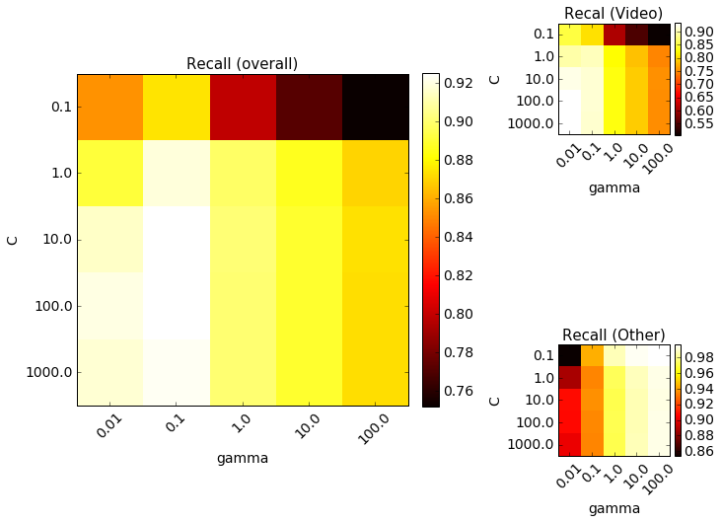

Figure 71 Exp. 5 Test 1 recall while varying C and gamma.

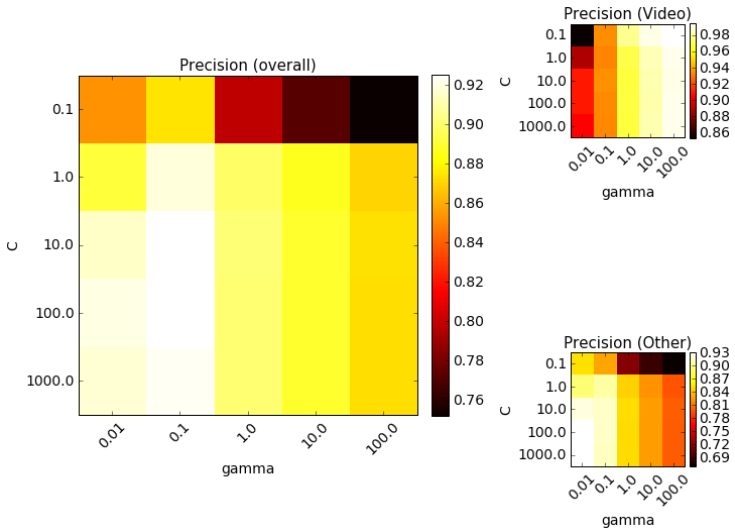

Figure 72 Exp. 5 Test 1 precision while varying C and gamma.

A.4.2. Test 2 – Payload size based features

In Test 2 the first ten individual payload sizes in addition to three statistical features based on these payload sizes were used. By inspection of Figure 74 and Figure 75 both C and gamma were set as 10. The classifiers performance can be seen in the confusion matrix and in the ROC curve in Figure 73. An overall accuracy of 92.56% was achieved which is as high as the benchmark test, i.e. Test 1 – All features, and a good indicator that the individual payload size features and the payload size based features are indeed good features.

Features: 13 features total.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylMean, PaylVar, PaylStd}

Table 24 Exp. 5 Test 2 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	9421	579	10000	94.21%	91.20%	
	Video	909	9091	10000	90.91%	94.01%	
	Total	10330	9670	20000			92.56%

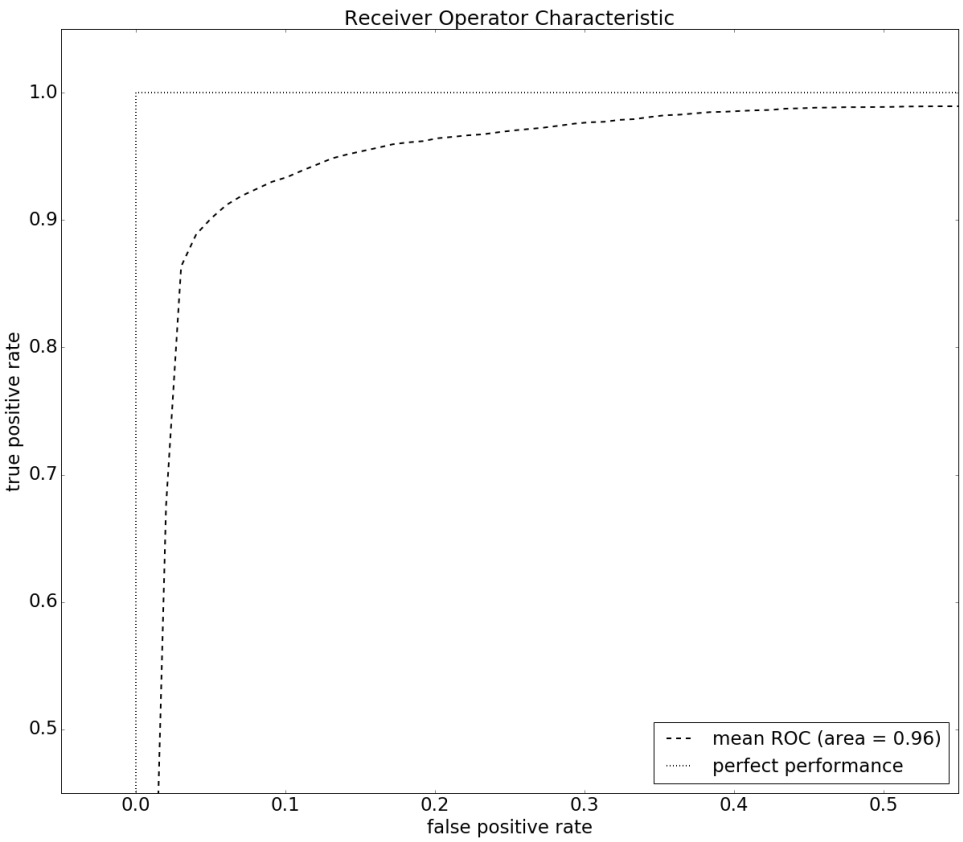

Figure 73 Exp. 5 Test 2 ROC curve.

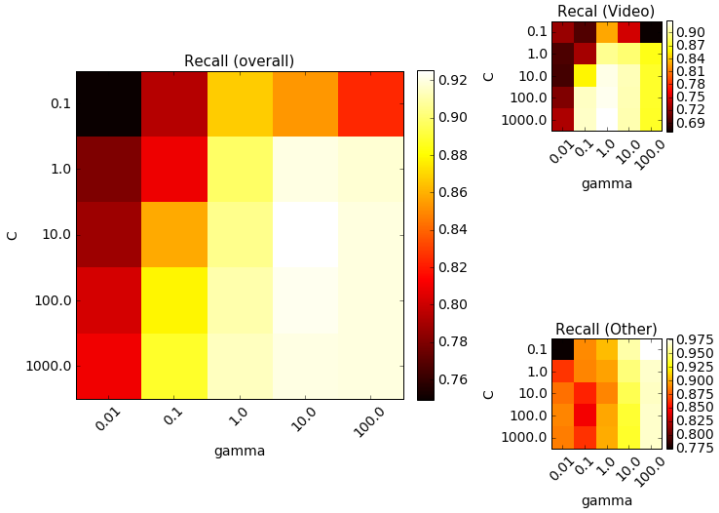

Figure 74 Exp. 5 Test 2 recall while varying C and gamma.

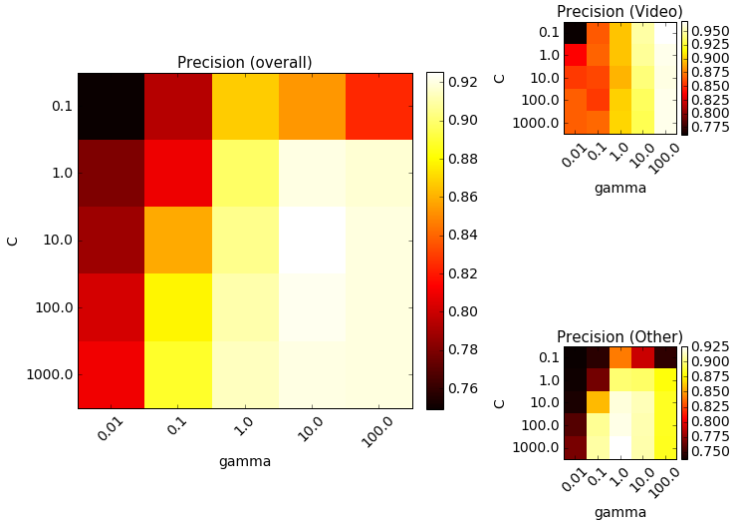

Figure 75 Exp. 5 Test 2 precision while varying C and gamma.

A.4.3. Test 3 – Payload based features

In Test 3 a total of 7 payload based features were used. By inspection of the heat maps in Figure 77 and Figure 78 the value for C was set to 10 and gamma was set to 100. The performance for the classifier can be seen in the confusion matrix and in the ROC curve in Figure 76. An overall accuracy of 91.00% was achieved. However, the recall regarding Video was only 87.14%.

Features: 7 features total.

Feature set = {PaylMean, PaylVar, PaylStd, DwnPaylRun200-20, BytesDwPayl-20, PosSumDwnPaylPkts-20, NrDownPaylPkts-20}

Table 25 Exp. 5 Test 3 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	9486	514	10000	94.86%	88.06%	
	Video	1286	8714	10000	87.14%	94.43%	
	Total	10772	9228	20000			91.00%

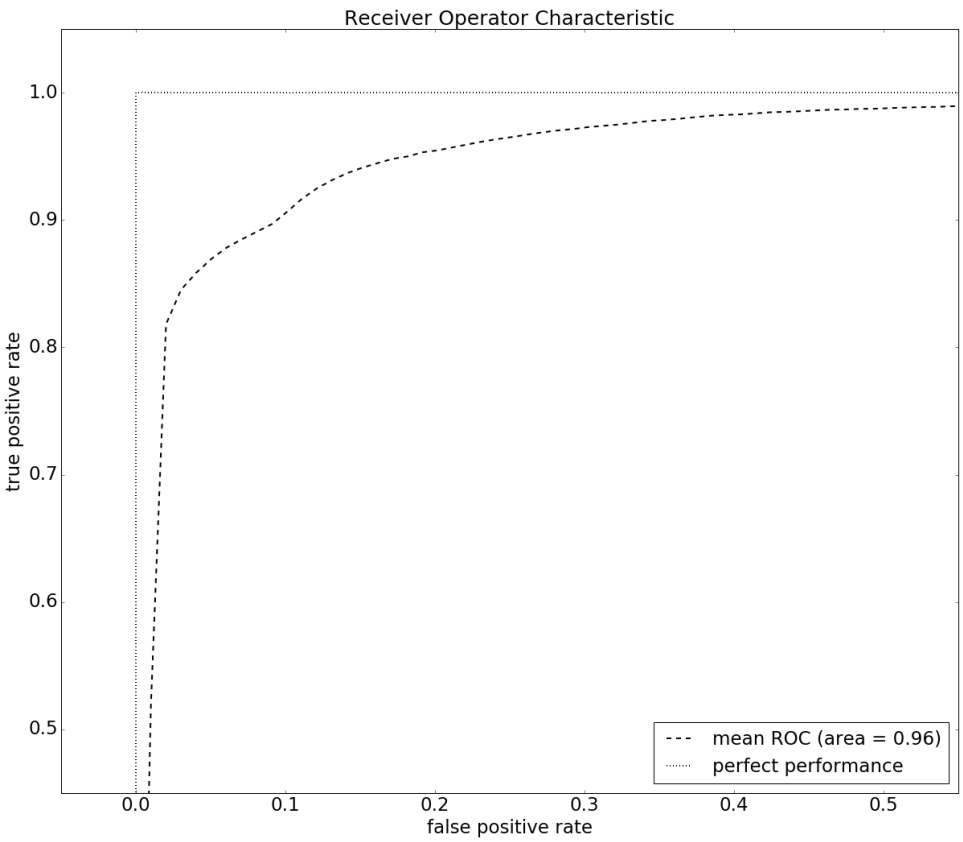

Figure 76 Exp. 5 Test 3 ROC curve.

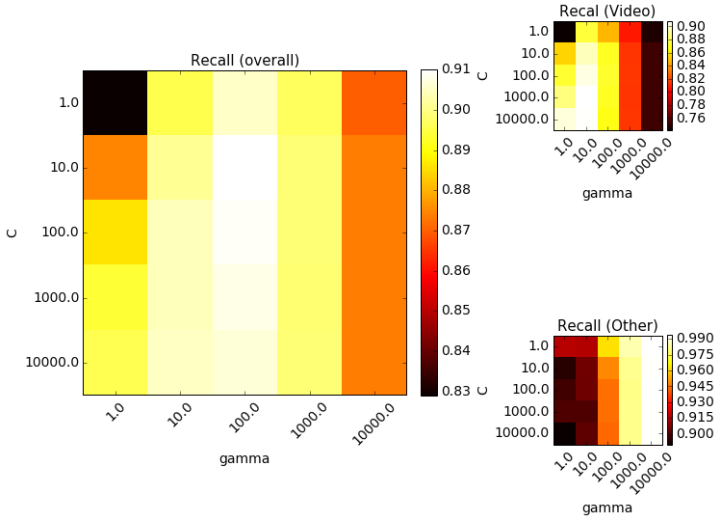

Figure 77 Exp. 5 Test 3 recall while varying C and gamma.

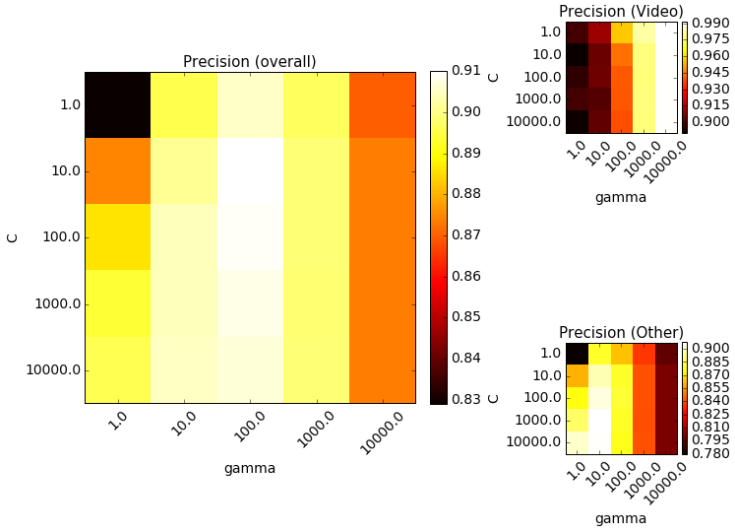

Figure 78 Exp. 5 Test 3 precision while varying C and gamma.

A.4.4. Test 4 – Payload sizes and payload based features

In Test 4 the combination of individual payload sizes and payload based features were used. The parameters C and gamma were chosen as 100 and 1 respectively by inspection of the heat maps in Figure 80 and Figure 81. The performance for corresponding model can be seen in the confusion matrix in addition to the ROC curve in Figure 79. The classifier achieved an overall accuracy of 92.55% which is about the same as the benchmark test. The recall and precision regarding Video is reasonable good as well, if compared to the benchmark test.

Features: 17 features total.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylMean, PaylVar, PaylStd, DwnPaylRun200-20, BytesDwPayl-20, PosSumDwnPaylPkts-20, NrDownPaylPkts-20}

Table 26 Exp. 5 Test 4 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	9304	696	10000	93.04%	92.13%	
	Video	795	9205	10000	92.05%	92.97%	
	Total	10099	9901	20000			92.55%

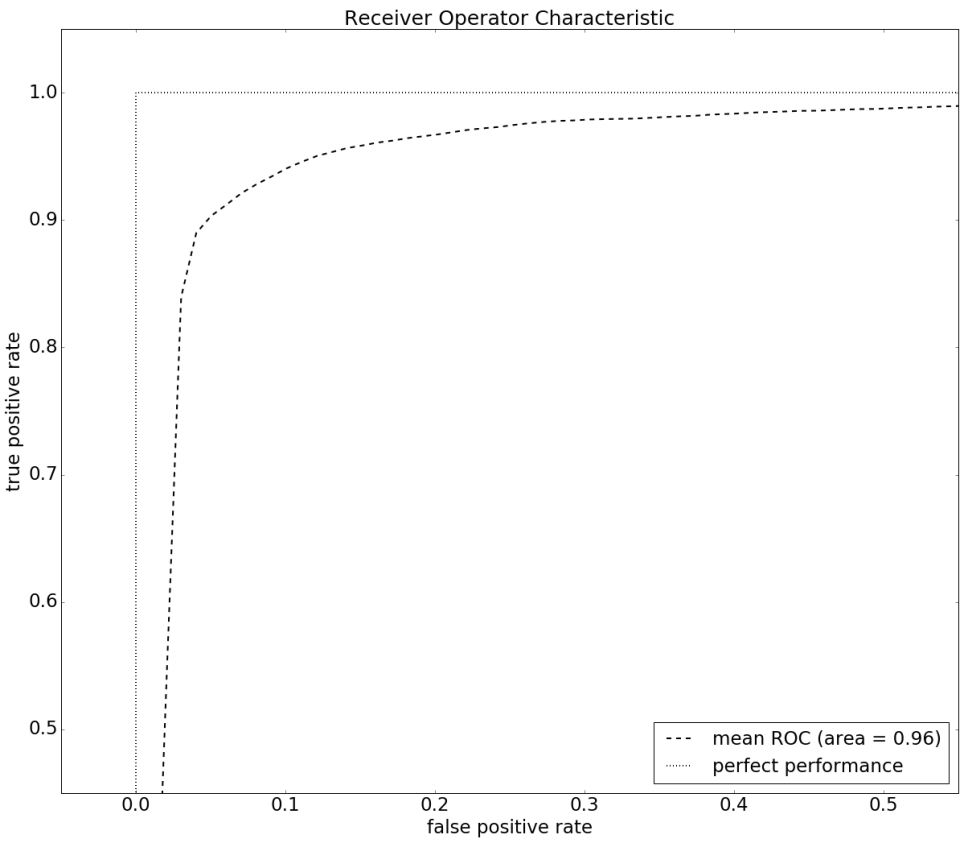

Figure 79 Exp. 5 Test 4 ROC curve.

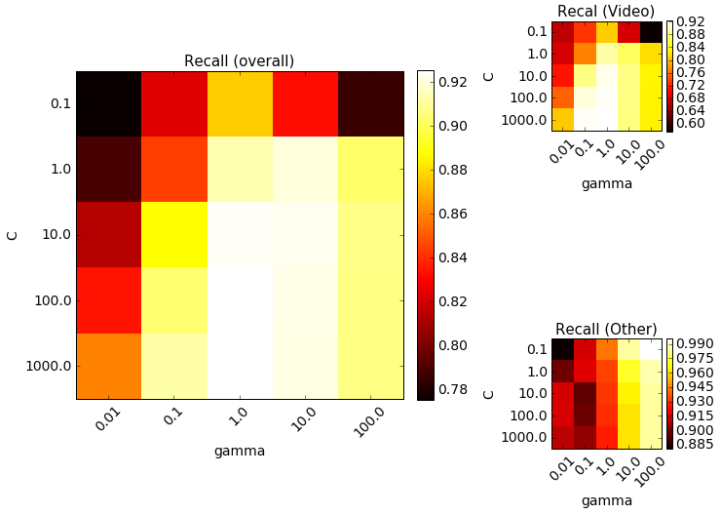

Figure 80 Exp. 5 Test 4 recall while varying C and gamma.

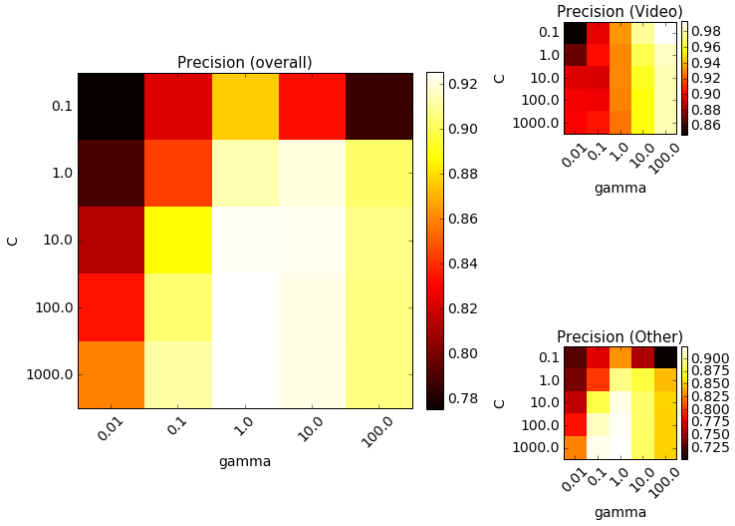

Figure 81 Exp. 5 Test 4 precision while varying C and gamma.

A.5. Experiment 6 – Alternative Video grouping SVC with RBF kernel using packet feature set: 20 first packets

Algorithm: SVC with RBF kernel

Initial feature set: Packet feature set

The features were computed based on the first 20 packets. Features were computed at packet indices 6, 8, 10, 12, 14, 16, 18, 20.

Sampled data set: *100k_100k*

Filtering specifics: The payload size was defined as >70 bytes. Only traffic flows with at least 10 packets were considered.

Further sampling: Asymmetric flows were removed. Because of how the SVC scales with more samples, recall Figure 19, the data set was further samples to include around 10000 Video flows and 10000 Other flows. Of the 10000 Video flows all but the flows of service "HTTP media stream" are redefined to belong to the Other class.

Filtered data set:

```
Number of Unique Flows   :  20000
Number of Unique Services:  129

Freq of Top 10 Internet services:

SERVICE                              |Count      | Share (%)
_____________________________________________________________
HTTP media stream                    |      6441 | 32.20
HTTP                                 |      3523 | 17.61
BitTorrent transfer                  |      1682 | 8.41
uTP                                  |      1147 | 5.73
Youku                                |      1028 | 5.14
BitTorrent KRPC                      |       670 | 3.35
DNS                                  |       619 | 3.09
RTMP                                 |       524 | 2.62
BitTorrent encrypted transfer        |       505 | 2.53
Kodi                                 |       416 | 2.08

Freq Video

SERVICE   |Count      | Share (%)
_____________________________________
Video     |      6441 | 32.20
Other     |     13559 | 67.80
```

A.5.1. Test 1 – Individual payload size features

In Test 1 only the individual payload size features were used comprising a total of 10 features. The parameter C was chosen as 1000 and gamma as 100 by inspection of the heat maps seen in Figure 83 and Figure 84. The performance for the corresponding classifier is seen in the confusion matrix in Table 27.

Features: 10 features total.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10}

Table 27 Exp. 6 Test 1 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	13292	267	13559	98.03%	94.71%	
	Video	743	5698	6441	88.46%	95.52%	
	Total	14035	5965	20000			94.95%

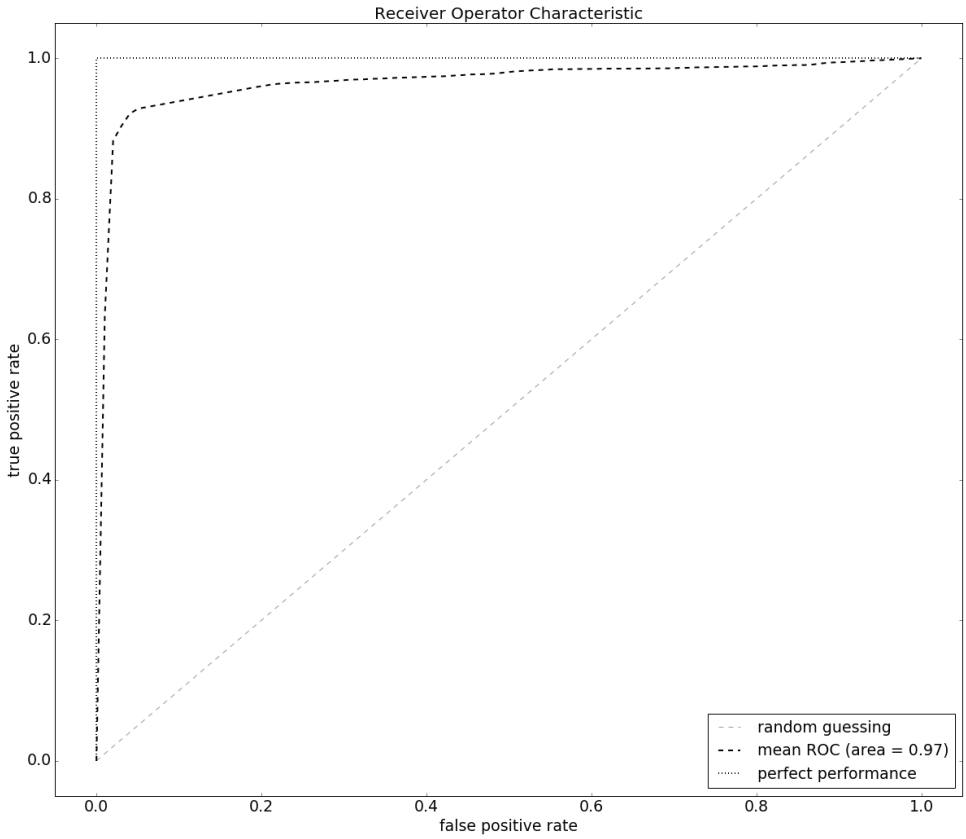

Figure 82 Exp. 6 Test 1 ROC curve.

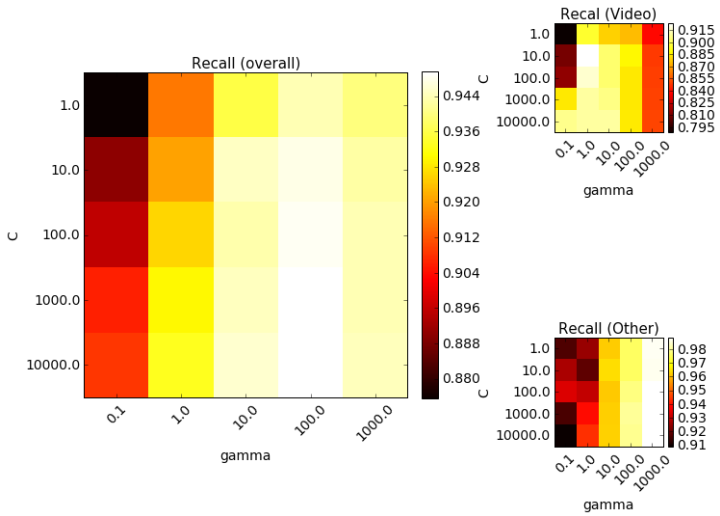

Figure 83 Exp. 6 Test 1 recall while varying C and gamma.

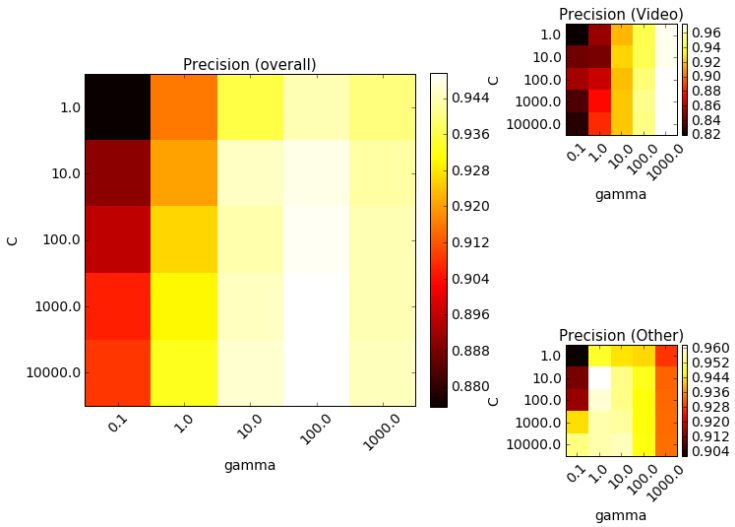

Figure 84 Exp. 6 Test 1 precision while varying C and gamma.

A.5.2. Test 2 – Payload size based features

In Test 2 the 10 individual payload sizes in addition to 3 statistical features based on the payload sizes were used. The parameters C and gamma were chosen as 10000 and 100 respectively by inspection of the heat maps seen in Figure 85 and Figure 86. The performance for the classifier can be seen in the confusion matrix in Table 28.

Features: 13 features total.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylMean, PaylVar, PaylStd}

Table 28 Exp. 6 Test 2 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	13302	257	13559	98.10%	94.56%	
	Video	766	5675	6441	88.11%	95.67%	
	Total	14068	5932	20000			94.89%

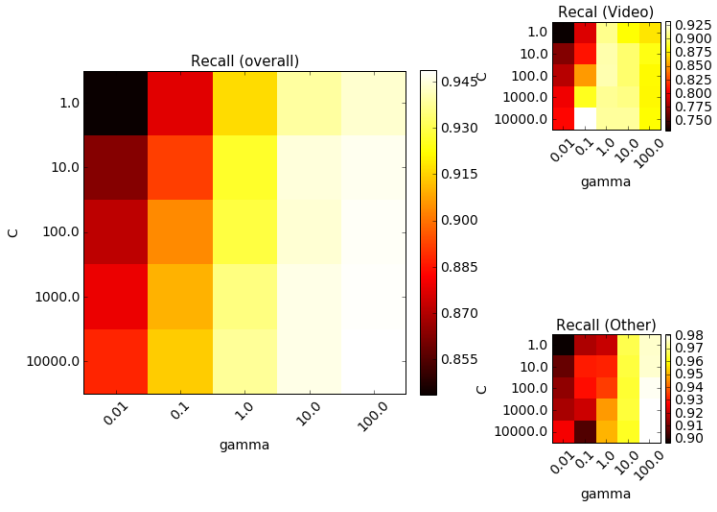

Figure 85 Exp. 6 Test 2 recall while varying C and gamma.

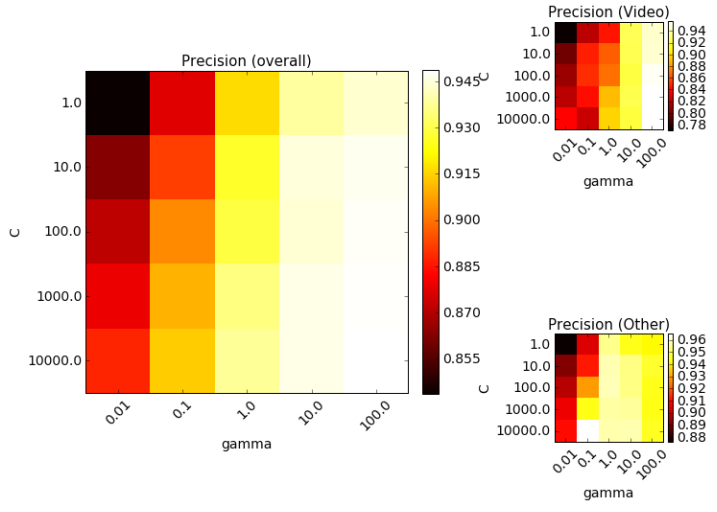

Figure 86 Exp. 6 Test 2 precision while varying C and gamma.

A.5.3. Test 3 - Payload sizes and payload based features

In Test 3 the individual payload sizes for 10 packets and 3 statistical features based on the payload sizes in addition to the payload based features were used. Both the parameters C and gamma were chosen as 10 after inspection of the heat maps in Figure 88 and Figure 89. The corresponding classifiers performance can be seen in the confusion matrix in Table 29.

Features: 17 features total.

Feature set = {PaylSize-1, PaylSize-2, PaylSize-3, PaylSize-4, PaylSize-5, PaylSize-6, PaylSize-7, PaylSize-8, PaylSize-9, PaylSize-10, PaylMean, PaylVar, PaylStd, DwnPaylRun200-20, BytesDwPayl-20, PosSumDwnPaylPkts-20, NrDownPaylPkts-20}

Table 29 Exp. 6 Test 3 confusion matrix including performance metrics.

		Predicted					
		Other	Video	Total	Recall	Precision	Accuracy
Actual	Other	13313	246	13559	98.19%	94.96%	
	Video	706	5735	6441	89.04%	95.89%	
	Total	14019	5981	20000			95.24%

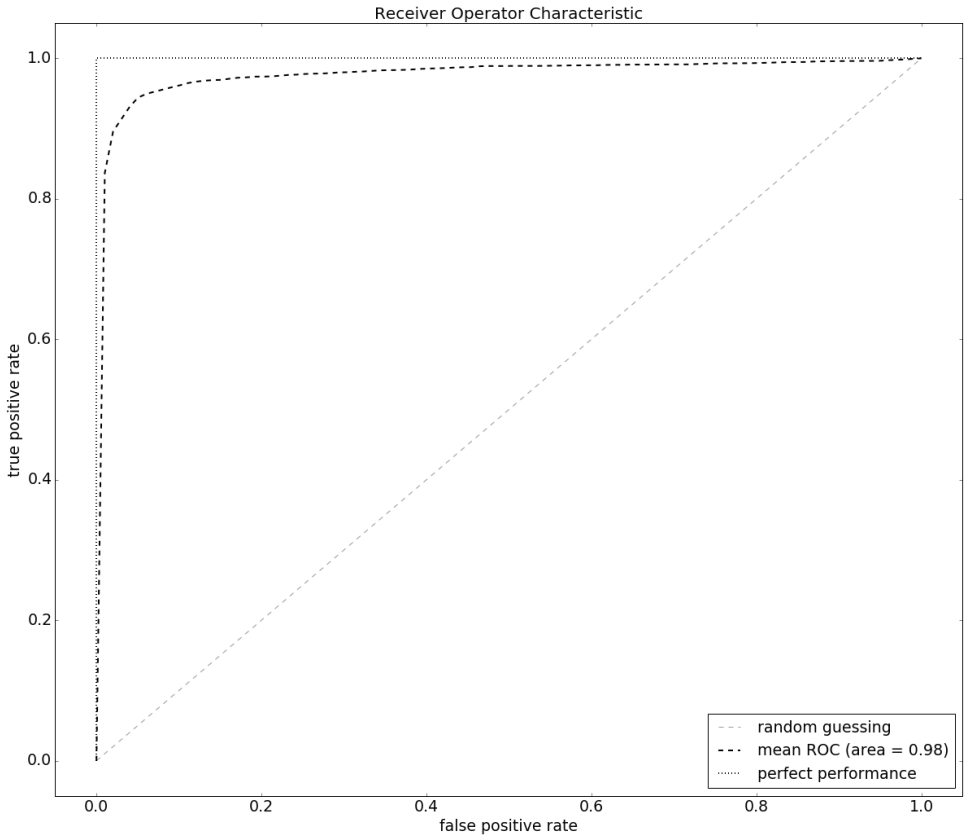

Figure 87 Exp. 6 Test 3 ROC curve.

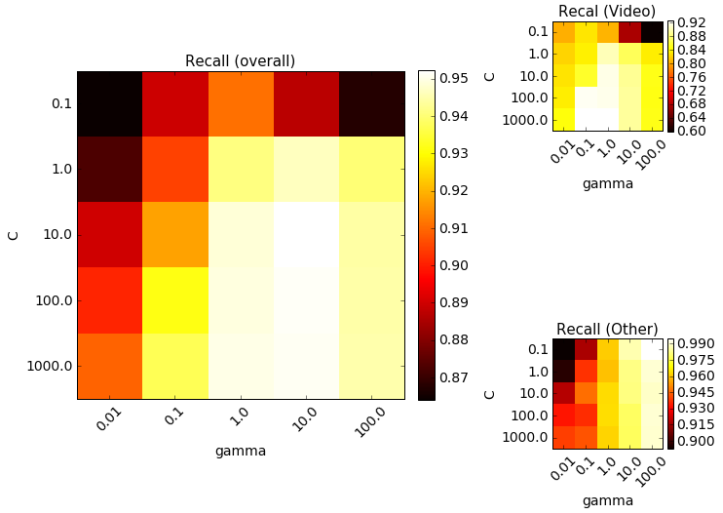

Figure 88 Exp. 6 Test 3 recall while varying C and gamma.

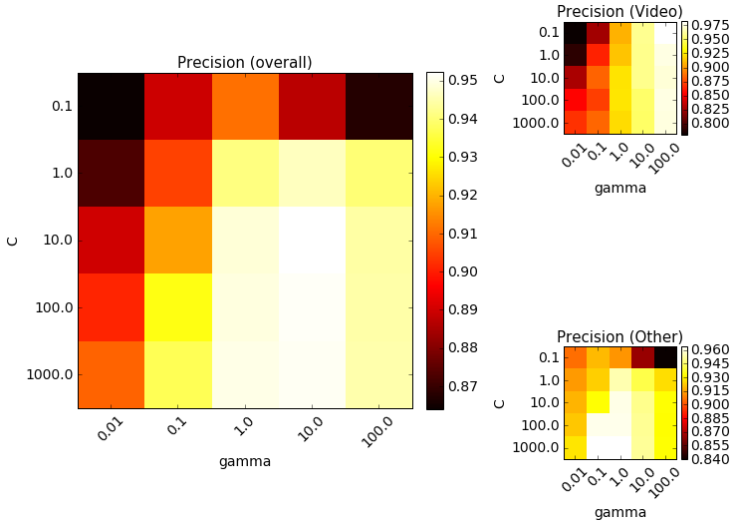

Figure 89 Exp. 6 Test 3 precision while varying C and gamma.