Using Support Vector Machines Organized in Ensembles to Efficiently Process Large Data Sets

Perfomance Measure of Ensamble SVMs on NSL-KDD Data Set

B. Baczyńska[1], L. Erhard[2], and M. Schaller[3]

[1] Poznan University of Technology (Poland), M.Sc. Computer Science, e-mail: beata.baczynska@estudiantat.upc.edu
[2] RWTH Aachen (Germany), M.Sc. of Data Science, e-mail: luisa.erhard@estudiantat.upc.edu
[3] Universität Leipzig (Germany), M.Sc. Computer Science e-mail: maximilian.schaller@estudiantat.upc.edu

December 12, 2021

ABSTRACT

. Intrusion Detection Systems (IDS) are operating in a challenging environment considering that they need to show a high accuracy as misclassification can endager important data, they must be fast as the environment may change quickly and they have to be capable of processing big amounts of data. Many machine learning models have been tried out over the time to meet these challenges.
. For example kernel Support Vector Machines are showing great performance on high feature spaces but don't score well with a rising number of data samples. Thus, this paper is aimed to use ensemble methods to see if the performance of a single kernel SVM can be kept while reducing the training time.
. The data set used in this work is a improved version of the *KDD'99* data set which was used for many years in IDS reseach but suffered from a high amount of redundant records.
. The experimental results show that ensemble methods can indeed keep the performance of a single linear SVM while decreasing the training time drastically.
. This work had a limited scope but its result motivate to further reseach with other kernels and ensemble methods to further improve the performance of the models.

. *Keywords* - Intrusion Detection – KDD'99 – NSL-KDD – Support Vector Machines – Ensemble Methods

1. Introduction

The internet brings not only benefits, it also endangers the security of the systems connected to it. There are static protection mechanisms like firewalls and we can improve the systems security through always working on the latest update. But it can also make sense to use an intrusion detection system (IDS) that monitors the network traffic for intrusions (attacks). With the rising number of attacks on computer networks the importance of using additional IDSs rises as well. IDSs can be divided into host-based and network-based and search either for misuse or for anomalies. Host-based IDSs concentrate on information of one computer system while network-based IDSs analyse network units. Searching for misuse means searching for known attack patterns. The anomaly technique on the other hand searches for unknown patterns in the systems behaviour [7].
The kind of data a IDS is confronted with, brings many challenges for the models applied to it. First, the prediction needs to be accurate, as a failure can lead to big security issues depending on how important the network or system is. Moreover, the models have to be fast as real-time prediction is necessary in such a quickly changing environment. Besides, they have to be capable of processing big amounts of data as the traffic volume is usually high and they must be able to catch complex relations [8]. This makes it a very challenging field for machine learning.
Methods that offer themselves in this environment because of its efficiency on data sets with many features are kernel SVMs. Through the transformation in possibly high-dimensional and non-linear feature spaces they provide flexibility and more possibilities for pattern recognition [8]. One main limitation of these methods is that they struggle with big amounts of data. This is what should be approached in this work. Using a random sample, instead of the whole data set, is less accurate because of the loss of much information. An ensemble of kernel SVMs could be a promising option to meet this limitation. This paper will present experiments with different ensembles of kernel SVMs where parameters will be tried out in several combinations and the role of the number of SVMs in the ensembles will be exploited. The assumption is that this approach will outperform normal linear SVMs in training time, prediction time and performance when working on the whole data set and the performance results will also be compared with other machine learning methods. The former version of the data set became popular when being the subject of a classification competition in 1999 that will be discussed in more detail in section 2.1.
The paper is organized as follows: As already mentioned section 2 describes the data set used in this work and some related

work is presented. Section 3 gives some theoretical background to the applied methods. The preprocessing, experimental setup and model selection is described in section 4. The results of the experiments are drawn in section 5 and section 6 has the objective to present the conclusion and to give place to a discussion and thoughts about future work.

2. Dataset Description and Previous Work

This Section describes first the data set that is used in this work and gives than an overview about previous work that is connected to the experiments presented here.

2.1. Dataset Description

KDD stands for Knowledge Discovery and Data Mining and is a platform that organizes data mining competitions. The data set *KDD'99* was subject of a competition from 1999 about intrusion detection. The task was to build a classification model capable of classifying a connection as either one of four attack (intrusion) categories or normal. For creating the data set Transmission Control Protocol (TCP) data was used that was recorded over nine weeks [7]. A TCP is a network protocol that defines how data should be exchanged between network components. A connection is a sequence of TCP packets between defined start and end times. In between these time steps data flows between two IP addresses under some protocol [1].
The connections are described by 41 features that can be discrete or continuous. *Duration* is an example of a continuous feature that describes the length of the connection. *Protocol_type* describes the type of the protocol and is therefore a discrete and categorical variable [1]. The connections are labeled as normal connection or attack. The attacks can be divided into four categories:

- DOS attack (denial-of-service attack) intends to make a network unavailable for the user through overtaxing the machine with many requests;
- U2R attack (User to Root attack) means that the attacker starts a normal user session and achieves access to the root through exploiting the systems weaknesses;
- R2L attack (Remote to User attack) intends to find gaps in the security system while the attacker acts from a remote machine;
- Probing attack means that the attacker scans a machine to find vulnerabilities that can be used to harm the system

[2][1].
The *KDD'99* data is divided into three different data sets: *Whole KDD*, *10% KDD* and *corrected KDD*. In the competition most of the participants used *10% KDD* for training and *corrected KDD* for testing [7]. The testing and training data are following different probability distributions because around a third of the attack types are only included in the testing data [2]. The whole KDD data set consists of about four million samples [6]. One major problem of the data set is the high amount of redundant samples. About 78% and 75% of the samples are duplicated in the train and test set [7]. Concerning the training data this can cause problems as the classifier is biased to the more frequent samples and less capable of recognizing new connections. When looking at the data for testing the redundant samples can give evaluation results in favor of models that perform better on already seen data [3]. To target this problem another data set was designed *NSL-KDD* which will be used for the experiments in this work. As test set *KDDTrain+.TXT* from [3] is used in this work which is here again splitted in a 20% : 80% split for the model selection. As test set *KDDTest+.TXT* from [3] is used. The important difference between *KDD'99* and *NSL-KDD* is that in *NSL-KDD* redundant samples were deleted. Also the *NSL-KDD* is much smaller as the original data set: It contains 125973 training records and 22544 test records (can be seen in 4.1). The preprocessing showed that the dataset is very unbalanced. In the training set 53% of the records were from the Normal attack class, 36% from DoS attack class, 9% from the Probe attack class, 0.8% from the R2L attack class and only 0.04% from the U2R attack type (See Figure number 1).
In the test set 43% of the samples were Normal connections, 33% from the DoS attack type, 13% from the R2L attack type, 11% from the Probe attack class and 0.3% of the samples were U2R attack connections (See Figure number 2).

Three different protocol types were present in the training set: icmp, tcp and udp. 50% of the icmp protocol incidents were Probe type of attacks, also 83% of the udp protocol events were not attack events. R2L attack type was only present when tcp protocol was used. (See Figure number 7).
All R2L attacks were carried out with tcp protocol, 94% of U2R attacks were carried out with tcp protocol and only 6% with udp. The rest of the attack classes as well as the normal events were met with each of the three protocols. (See Figure number 8).
The relationship between the type of attack and flag was investigated. For Denial of service attack S0 flag was visible 75% of time. For Probe attack the SF flag was visible 50% of the time (See Figure number 9).
Remarkable is also that new types of attack were present in the testing set. The train set consists only of 23 types of attack while the testing set includes 38 types. 17 new types of attack were present in the testing set and 2 types of attack that were in the training set have never appeared in test one. This different distribution of attack types is present to prepare the models for unseen attack types in real applications.

2.2. Previous Work

Both the original *KDD'99* and the derived *NSL-KDD* data sets are well- known in IDS research and have therefore been used many times for classification. For this reason here will only be mentioned works that are compared with the here presented experiments, use similar methods or have an importance in

the history of this data. First, worth to mention is the winning entry of the competition in 1999. There they used a decision tree ensemble on the *KDD'99* data set. But they also deleted duplicate entries as done for *NSL-KDD*[11].
A paper that will play a bigger role in this work is [6]. There they designed a shallow linear model (shallow neural network with linear activations) with a feature transformation using kernel algorithms. They compared this design with models (Linear and RBF SVMs, Multilayer Perceptron (MLP), Gradient Boosting Machine (GBM), Random Forest, AdaBoost, Multinomial Logistic Regression and Convolutional Neural Networks) and wanted to show that their design competes in respect to prediction performance and outperforms the non-linear models time-wise. Considered were three data sets of which one is *NSL-KDD*. As will be showed in 4.1 we will use the same preprocessing to be able to compare our results to the ones obtained in this paper.
Another paper that presents interesting results is [7]. It uses as well the *NSL-KDD* data set and experiments with different kernel SVMs in the search of the kernel and parameters that deliver the best results. Their results show that the RBF kernel SVM performs better than the other kernel choices in terms of the detection rate.

3. Theoretical Background

In the first part of this section a short introduction into support vector machines for classification tasks is given. The section continues with an overview about ensembling methods.

3.1. Support Vector Machines

Assuming a set of points (observations) χ is given which should be classified into two distinct classes. SVMs are supervised classification methods which try to find a hyper plane in the feature space which separates these two classes. If the data is linearly separable the method aims to maximize the margin between the two classes to gain the highest generalization. But in many cases the data is not linearly separable. In these cases the use of a non-linear feature map

$$\phi : \chi \to H$$

allows to transfer the points into a higher dimensional space (Hilbert space) H in which the points might be linearly separable. As SVMs can be transformed to a dual form (equivalent to the primal form) in which the data only appears in form of a inner product the kernel trick can be applied. The trick is to substitute the scalar product of data by a kernel function

$$k : \chi \times \chi \to \mathbb{R}.$$

The kernel function can be understood as a similarity measure between the observations. In this way there is no need to know the coordinates of the points in the feature set only its pairwise inner product matters. Instead of calculating a inner product in a possibly infinite dimensional feature space only the kernel function has to be evaluated on the input space χ. In this paper only the linear kernel function is used which is defined as

$$k(x, y) = x^T y.$$

The dual problem of a soft SVM (allowing points to lay in the margin or on the wrong side of the hyperplane) with kernel function has the form

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

which is subject to

$$0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_i \alpha_i y_i = 0.$$

The parameter C decides about the tradeoff between model complexity and prediciton accuracy because it influences the error term. Decreasing the C value gets along with an decreasing margin which might lead to overfitting to the data [9].
Multi-class classification is also possible and can be implemented in different manners. The one chosen in this work is the one-versus-one approach. This means for the five classes {Normal, DOS, U2R, R2L, Probing} there are $5 * 4/2 = 10$ classifiers which train each two classes: {Normal-DOS, Normal-U2R, Normal-R2L, Normal-Probing, DOS-U2R, DOS-R2L, DOS-Probing, U2R-R2L, U2R-Probing, R2L-Probing}. This is included by default in the *sklearn.svm.SVC* function [10]. The predicted class is then the one that received the most votes from the set of binary classifiers. This approach is noramlly slower than one-versus-the-rest which would only give five classifiers in this case. But the one-versus-the-rest approach would use the whole data set here five times while the one-versus-one approach involves each time only a small subset of the data. That is an advantage for kernel SVMs as they are not well handling big amounts of data [10].

3.2. Ensemble Methods

The idea of ensemble methods in machine learning is to combine several weaker learning algorithms to a stronger one. In the case of SVMs the ensemble method is a supervised learning algorithm. The ensemble type that will be used here is Bootstrap aggregating, also called Bagging. For Bagging each single model of the ensemble is trained on a randomly created subset (with overlapping or without) of the training set where all subsets union to the whole training set [13]. The prediction is then made through aggregating the predictions of all sub-classifiers. There are two main aggregation methods: Majority vote which is used here gives the class that was predicted the most often by the sub-classifiers. The majority voting is also called hard voting. The soft voting method assumes that the sub-classifiers give back probabilities for the classes. Based on these the Probability sum method sums up these probabilities and predicts the one with the highest sum [12]. The random forest algorithm for example uses Bagging to combine decision trees.

Another type is called Boosting. It builds the ensemble by concentrating in each next step on the previously miss classified records. A famous example for Boosting is the Adaboost algorithm [13]. Both, bagging and boosting are reducing variance. Boosting can also reduce bias, but it's easier to over-fit, while bagging method can prevent from the over-fitting problem [4].
A further ensemble type is stacking where several different classifiers are used in parallel and there outputs are fed into a final meta-model that does the prediction *sklearn.svm.SVC*.

4. Preprocessing and Experimental Setup

This Chapter explains the surroundings and setup of the experimentation. Therefore, Section 4.1 gives more detailed insights on how the data, described in Chapter 2.1, is preprocessed. Furthermore, Section 4.2 introduces the experimental setup, containing an overview of the used hardware, the evaluation metrics and the limitations of the scope of the experiments. Finally, Section 4.3 provides a brief description of the implemented architecture.

4.1. Preprocessing

The following changes were made to the NSL-KDD dataset: Attacks were divided into 4 groups: DOS attack, U2R attack, R2L attack and Probing attack. Including normal (non attack) events, 5 classes were received. These classes were transformed to numerical type (0-4). Categorical variables were transformed by one-hot encoding methods to binary ones. Numerical variables were scaled to range between 0 and 1. At the end received 122 features and one output.

4.2. Experimental Setup

This Section describes the experimental setup used during the experiments. It contains the used hardware, the metrics and the limitations.

4.2.1. Hardware

The training of the Ensamble SVM runs on a Macbook Pro 2020 that uses a 2 GHz Quad-Core Intel i5 Processor and 16GB of memory.

4.2.2. Metrics

In this paper there are several different metrics used to evaluate and compare the performance of the implemented Ensamble SVM and to find the best fitting hyper parameter. As one of the fundamental goals of this paper is to build an efficient approach on using SVMs on large data sets. Therefore, it is essential to measure the time the algorithm needs to perform a complete training using the whole data set. In the case of this work, the training time t is the difference between the time at the beginning of the training and the end of the training: $t = t_{end} - t_{start}$

For comparing the results of the approach used in this paper to the work of M.Lopez-Martin et al. [8], apparently it is necessary to use the same metrics. Thus, this paper utilizes the accuracy score and the weighted average of F1, recall and precision score with functions provided by the Scikit-learn [10] Python library.
Because of the huge imbalance between classes: the majority class normal (no attack) makes up 53% and the minority class U2R (attack category) corresponds only to 0.04%, it was decided to not find accuracy as an important measure. When the data set is imbalanced, for a model that is always choosing the majority class, accuracy is just equal to the proportion of that class and it is not possible to check performance. Two important measures are crucial to evaluate performance of models trained with an imbalanced data set. It's precision and recall, both are based on confusion matrix content. Precision is telling the ratio between correctly detected instances of class A and all true class A instances.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Recall is showing the ratio between correctly detected instances of class A and all instances detected as class A.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

F1 score measures the trade-off between precision and recall, so the F1 score is the measure that was finally chosen as the decisive score for evaluating the best parameters in the model selection process.

$$\text{F1-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.2.3. Limitations

Due to a lack of time and resources there are a plenty of limitations in the scope of this work. As first only the linear kernel is used during experimentation, which is sufficient as this work only focuses on proving, that it is possible to have a machine learning approach that uses SVMs, but still can reach considerable results compared to both, other SVMs and also, in general, other machine learning approaches. Furthermore, for the same reasons, the scope of the evaluation of the hyperparameters is very limited and restricted to a small selection, that is described in section 4.2.4 Another important thing to mention is, that the training time is compared to a SVM implemented and executed on the same machine as the ensemble approach, but as the available resources only provided a normal working laptop, it is not possible to ensure that the utilization of the computational power was exact the same for all experiments, and, thus, the measures of duration has to be seen with caution, and might differ a bit in a better setup.

Finally the performance evaluation is limited in comparing the outcome with the results in the work of M.Lopez-Martin et al. [8] and with the single SVM obtained here.

4.2.4. Hyperparameter

A crucial part on evaluating if a machine learning algorithm is suitable to a problem is to find the best fitting hyperparameter to reach the best possible solution. Unfortunately, the search for those was very limited by time and amount of data. As already mentioned, the training data has been split into 80% for training and 20% for the validation set. For the parameters, there were two main ideas to split the training: first evaluate a small subset of N number of SVMs used in the ensamble and evaluate a bunch of different cost paremeters C on the validation set. Each for the different SVMs in one ensemble is, in first place, trained and evaluated on the same C. The following table provides an overview of the researched parameters.

Hyperparameter	Values
N	$\{5, 7, 10\}$
C	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$

Table 1. Overview of investigated hyperparameter

By inference from the paper [8] and subsequently running a bunch of random tests, we arrived at this parameter choice.
To find fitting parameters a grid search has been performed on a small subset of those parameters. Thus, for the training data was split into N buckets of the same size and every single SVM received one of the buckets to train. The performance of each particular run was then measured on the validation set. Thereafter, in a greedy manner, the best number of SVMs, concerning the weighted average F1 score has been selected and as well the best C parameters.

4.3. Design of the Ensemble SVM

To build the ensemble SVM, N SVMs have been created. Besides the cost parameter, that has been evaluated as described in Section 4.2.4, all are using a linear kernel and also class weights. As mentioned in 2.1 the amount of data for the different labels (attack categories) is very unbalanced. To approach this problem the *sklearn.svm.SVC* function offers a parameter called *class_weight*. It is a dictionary that assigns one positive value for each attack category. These values are then multiplied each with the C parameter to have a different strictness for different classes [10].

Training

For the training the data has been randomly shuffled and split into N buckets of the same size. To achieve comparable results between the different runs, the shuffled data was stored and reused for every different run of the hyperparameter evaluation. Afterwards, a certain bucket was assigned to each of the N machines and than every machine was trained in parallel.

Prediction

In both the validation stage, as well as in the test stage, majority voting has been used for predicting. Therefore, every single data point of the concerning set has been shown to each of the SVMs in the ensemble. Then, every SVMs made its own prediction, and, thus, the class label that was predicted the most has been chosen as the final outcome.

5. Experimental Results

The goal of the first experiment was it to find in the model selection process (training and validation set) the best choice for a cost parameter and number of SVMs. The found model should then be compared to other machine learning methods based on its performance and with a single SVM ran on the same computer to compare the training times.
The first step was to find the best number of SVMs for the ensemble method. Compared were the numbers: five, seven and ten. In table 2 the average performance on the validation set is compared together with the training times.

Further experiments will be executed based on SVM ensembles consisting of five SVMs because five as number of SVMs showed the best F1-score.
Based on this the best suiting value of the C parameter should be chosen. The following set of possible values was considered $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$. Table 3 shows the performance of five ensembles of five machines based on the different values of the C parameter.

Looking at the scores the ensemble with cost parameter 100 has the best F1-score. The two figures 5 and 6 of the F1-score visualize the relationship between the chosen values and the F1-score. The first figure 6 shows how the F1-score changes given a cost parameter with value 100 depending on the number of SVMs in the ensemble. The F1-score is the highest for five machines and decreases for seven and even more for ten machines. The second figure 5 on the other side shows the change of the F1-score given an ensemble of five machines depending of the choice of the cost parameter. The F1-score rises with growing C-value but from 100 on it stays more or less the same. The ensemble with cost parameter 1000 has a very similar F1-score like the ensemble with C-value 100, but considering the figure 3 of the training time depending on the Cost parameter given an ensemble with five SVMs it can be seen that the training time rises significantly between C-values 100 and 1000. So the choice of C-value 100 is the better one.
Having chosen the best model based on the validation set this model should at first be compared with a single SVM with the same Cost parameter. Table 4 compares the two models with each other on the test set (training time was taken in training on training set).

It can be seen that the two models are having quite similar performance results: The Single SVM has a F1-score of 73.4% while the ensemble has a score of 72.7%. Moreover the Single SVM has a Recall of 76.4% and the ensemble has a Recall of

Number SVMs	C-Value	Accuracy	F1	Recall	Precision	Training Time
5	123.5	0.9816	0.9810	0.9816	0.9805	155.31
7	123.5	0.9809	0.9802	0.9809	0.9798	186.42
10	123.5	0.9796	0.9790	0.9796	0.9786	106.54

Table 2. Averaged Scores grouped by Number of Machine

Number SVMs	C-Value	Accuracy	F1	Recall	Precision	Training Time
5	1000	0.9913	0.99138	0.9913	0.9916	981.85
5	100	0.9913	0.99141	0.9913	0.9916	196.50
5	10	0.9909	0.99104	0.9909	0.9913	44.78
5	1	0.9899	0.99003	0.9900	0.9903	18.93
5	0.1	0.9866	0.98681	0.9866	0.9872	13.44
5	0.01	0.9806	0.98089	0.9806	0.9815	14.34
5	0.001	0.9740	0.97546	0.9740	0.9669	19.14
5	0.0001	0.9816	0.97036	0.9816	0.9805	34.22
5	0.00001	0.9545	0.95119	0.9545	0.9484	74.60

Table 3. Scores for SVM Ensembles by Cost Parameter

Method	C-Value	Accuracy	F1	Recall	Precision	Training Time
Ensemble SVM (5 SVMs)	100	0.7559	0.7270	0.7559	0.7835	196.50
Single SVM	100	0.7642	0.7336	0.7642	0.8080	16856.33

Table 4. Comparison Ensemble SVM with Single SVM

75.6%. Finally the Single SVM has a Precision of 80.8% while the ensemble has a Precision of 78.4%. The greatest difference can be seen in the training time. The Single SVM needs around 85 times of the training time the Ensemble needs. This relation can also be observed looking at the figure 4 that shows the training time depending on the number of machines given a cost parameter of 100. The training time decreases strongly when increasing the number of machines. After reaching the number five the training time doesn't decrease much more.

Method	Accuracy	F1	Precision	Recall
Log. Regression	0.660	0.607	0.657	0.660
Linear Kernel	0.756	0.729	0.762	0.756
RBF Kernel	0.806	0.806	0.813	0.806
Random Forest	0.739	0.691	0.771	0.739

Table 5. Results from the Paper [8]

Looking at the performance scores in table 5 of different methods from the paper [8] it can be seen that the performance of SVMs with linear kernel for both methods - ensemble and single - is similar to the performance of the linear SVM in the paper. These methods can be considered as better choice for this problem than logistic regression and random forest. The result described in the paper for SVMs with RBF Kernel is still the best one.

Unfortunately the training times in the paper can not be compared with the one found here as this depends strongly on the used hardware. However, by comparing results for ensemble SVM method and single SVM, with similar final performance, it can be said that it's highly recommended to use the ensemble approach for this problem as the training time was around 85 times lower.

6. Conclusion

6.0.1. Discussion of the Experiments

The expectation was that the ensemble method would have a better performance than the single SVM, as this is often the case for decision tree ensembles. However the performance doesn't improve, it stays more or less the same. But through training each SVM of the ensemble on a different part of the data set it was gained diversity in between the SVMs. Also the ensemble methods help prevent the model from overfitting to the data because even if the individual SVMs overfit the ensemble might correct that in total.

The training time was reduced drastically through the Ensemble method which is a big advantage for IDSs because through fast changes in the network environment constant periodic training is required and a shorter training time improves therefore the effectiveness of the IDS [8]

The expectation for the prediction time is that the ensemble method decreases the time needed for prediction because the number of support vectors increases linearly with the number of training examples and the prediction time is proportional to the number of support vectors [5].

6.0.2. Critical Assessment

The original plan was to execute two different experiments, the one presented here and the second was a comparison of the presented ensemble method of non-overlapping data that should be compared with an ensemble method of overlapping data sets. So the different SVMs should be trained on overlapping data. Because of the limited time that was given for the experiments it was not possible to finish the latter. As an example the experiments of the non-overlapping data took more than 30 hours.
Data was split randomly to make sure that the order of the data set is not kept, but the occurrence of at least one sample from each class was ensured manually. Through the random split the obtained results are unique and could be different when the experiments would be repeated.

6.0.3. Possible Future Work

In this work only the linear kernel was used but there are much more interesting kernels. Through comparing the results gained here it could be seen that the RBF kernel SVM used in the paper [8] performed better than the ensemble. So it would be interesting to use the RBF kernel for example in an ensemble method. Also because of lack of time and computing resources only a small set of C parameters and number of SVMs were tried out for the experiments. The method for parameter selection used here was a kind of grid search algorithm. One option to improve would be to take wider subsets of hyperparameters for grid search when the resources are less limited. But as grid search is very demanding it would be better to use a more advanced approach for parameter selection. The majority voting between Support Vector Machines could be changed to the sum of probability voting which was explained in 3.2.
Also the method for dividing the data set for each SVM could be different. We could continue with overlapping data set, choosing data sets completely randomly (with repetitions).
Subject to further research can also be if a bigger or several validation sets can improve the performance of the ensemble method. In this paper 20% of the training data was used only for validation (model selection). Instead it would be also an idea to take for example 20% of each data subset (as many as SVMs) as validation sets for the corresponding SVM. It could produce more diversity in between the SVMs.
To control the number of support vectors and margin errors directly it could be useful to use the ν parameter instead of the C parameter for the SVMs. Because the higher number of support vectors can lead to worse final performance, it would be beneficial to have a direct influence for that number. Also an interesting experiment will be to compare ensemble SVM method with single SVM that in total are having the same number of support vectors.
On the test data was also tried out to see if different combinations of C parameters (for the ensemble SVM method) would improve the performance. Instead of using the same value of C parameter for all five SVMs it was tried out to combine the three best C's values (10, 100 and 1000) in different combinations. This led indeed to better results in some cases on the test data and motivates to research on that in future experiments. A second validation set could be used to compare the performance of the different models each with another distribution.

References

[1] Cup-99 task description. http://kdd.ics.uci.edu/databases/kddcup99/task.html. Last accessed on 2021-12-09.

[2] Handling structured imbalanced datasets: Kdd cup 1999. https://www.analyticsvidhya.com/blog/2016/10/investigation-on-handling-structured-imbalanced-datasets-with-deep-learning/. Last accessed on 2021-12-09.

[3] Search unb. https://www.unb.ca/cic/datasets/nsl.html. Last accessed on 2021-12-09.

[4] What is the difference between bagging and boosting. https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/. Last accessed on 2021-12-09.

[5] Bottou, L., Weston, J., and Bakir, G. Breaking svm complexity with cross-training. In *Advances in Neural Information Processing Systems* (2005), L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17, MIT Press.

[6] Ch.Ambedkar, V. K. B. Detection of probe attacks using machine learning techniques. *International Journal of Research Studies in Computer Science and Engineering 2* (2015), 25–29.

[7] Hasan, M. A., Nasser, M., and Pal, B. On the kdd'99 dataset: Support vector machine based intrusion detection system (ids) with different kernels. *International Journal of Electronics Communication and Computer Engineering 4* (07 2013), 1164–1170.

[8] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., and Lloret, J. Shallow neural network with kernel approximation for prediction problems in highly demanding data networks. *Expert Systems with Applications 124* (2019), 196–208.

[9] Muller, K.-R., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks 12*, 2 (2001), 181–201.

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

[11] Pfahringer, B. Winning the kdd99 classification cup: bagged boosting. *SIGKDD Explor. 1* (2000), 65–66.

[12] Stork, J., Ramos, R., Koch, P., and Konen, W. Svm ensembles are better when different kernel types are combined. 191–201.

[13] Ying, X. Ensemble learning. https://www.researchgate.net/profile/Xu-Ying-7/publication/262369664_Ensemble_Learning/links/0046353762bbf908ea000000/Ensemble-Learning.pdf, Last accessed on 2021-12-0.

Appendix

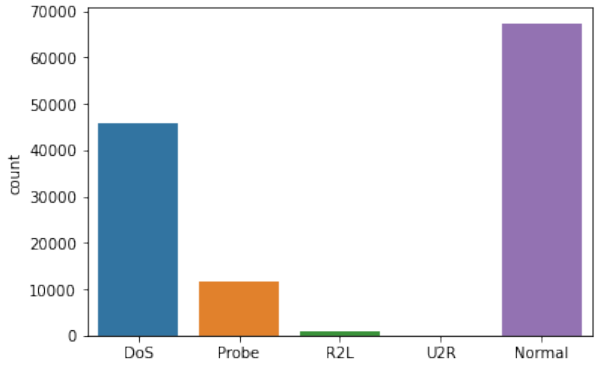

Figure 1. Samples distribution by attack class in the train set.

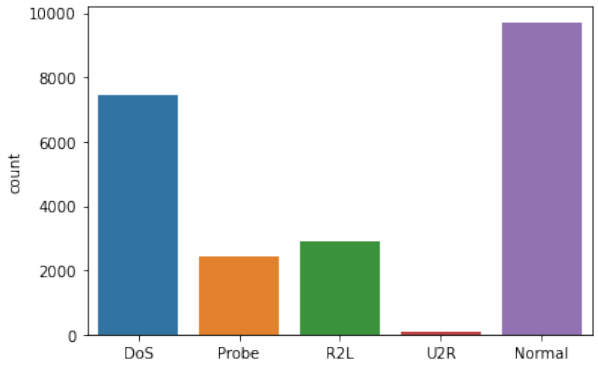

Figure 2. Samples distribution by attack class in the test set.

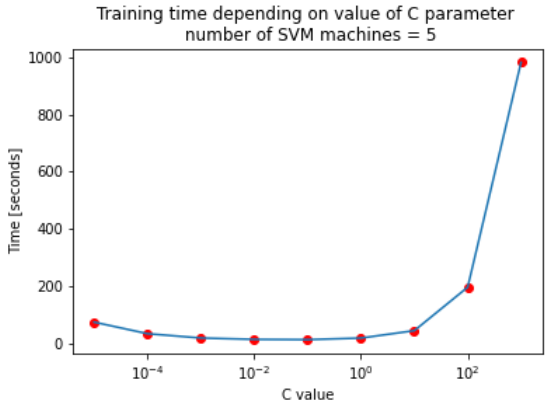

Figure 3. Training Time relation to the value of the C parameter

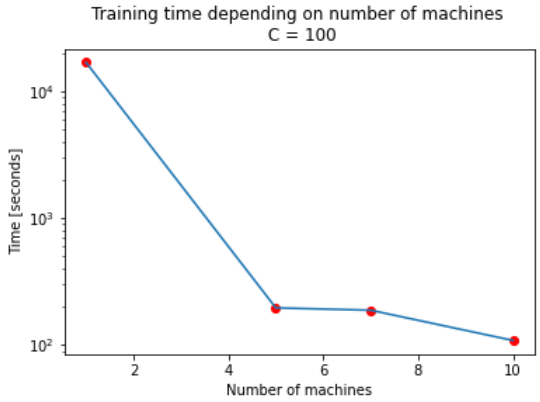

Figure 4. Training Time relation to Number of Machines

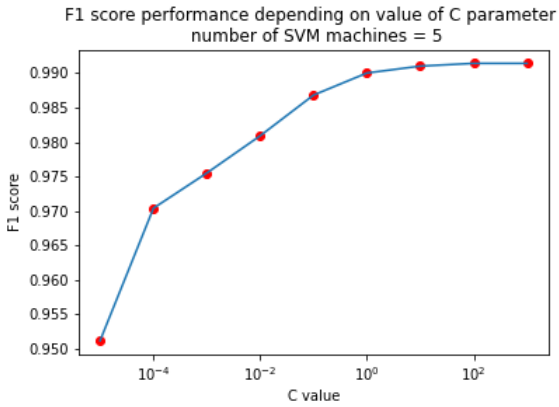

Figure 5. F1-score relation to the value of the C parameter

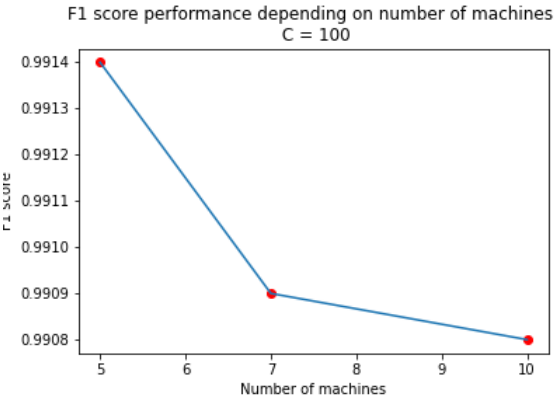

Figure 6. F1-score relation to Number of Machines

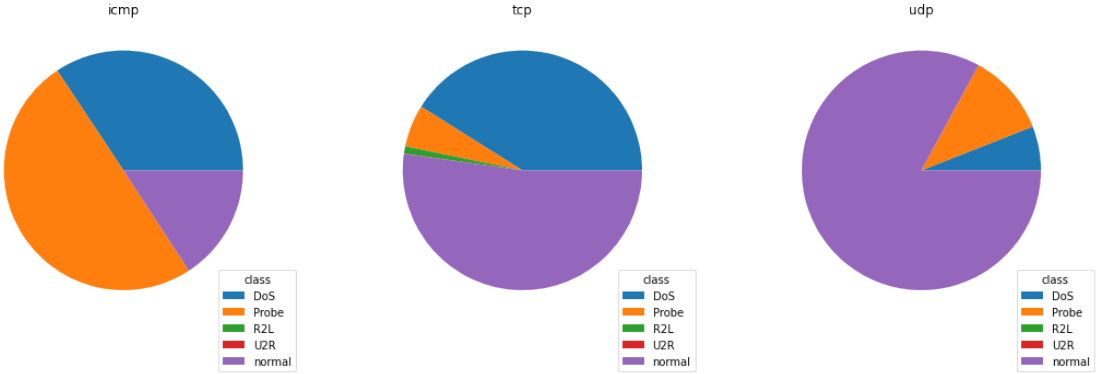

Figure 7. Attack classes by protocol types

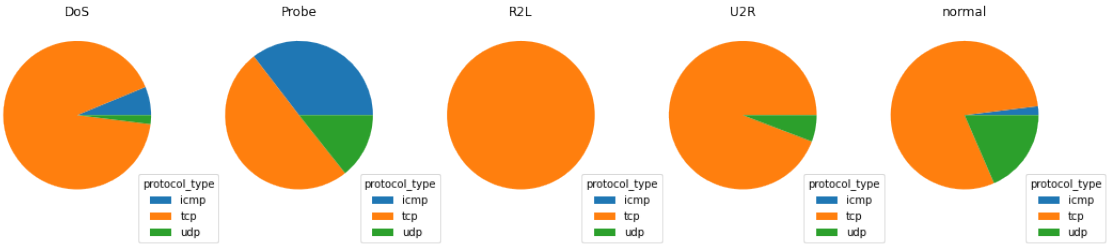

Figure 8. Protocol types by attack classes

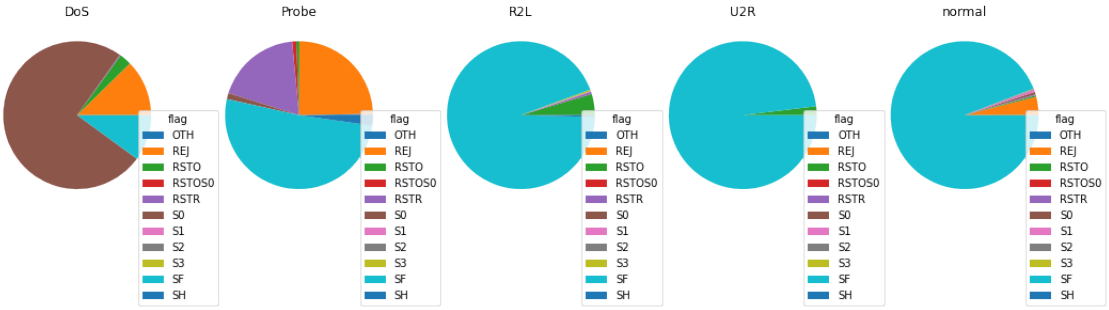

Figure 9. Flags by attack classes