A Biologically-Inspired Approach to Network Traffic Classification for Resource-Constrained Systems

Brian Schmidt, Dionysios Kountanis, Ala Al-Fuqaha
Computer Science Department
College of Engineering and Applied Sciences
Western Michigan University
Kalamazoo, Michigan, USA
{brian.h.schmidt, dionysios.kountanis, ala.al-fuqaha}@wmich.edu

Abstract*—Internet traffic classification has been studied widely in recent years, and many machine learning approaches have been applied to it. Internet traffic classification has increased in relevance in recent years because of its potential applications in the business world. Information about network traffic has many benefits in network design, security, management and accounting. Internet traffic classification is especially important to the adaptive networks often used in cloud computing, which must use data gleaned from the network to adjust to network conditions on-the-fly. This information is most easily collated from the huge amount of information going through a modern network with machine learning algorithms, which adjust themselves to the conditions of the network. In previous research, Artificial Immune System (AIS) algorithms have been used to classify malicious and benign network traffic in support of Intrusion Detection Systems [1]. Because of their versatility and their low sensitivity to the values of the input parameters, we are motivated to explore the value of using AIS inspired algorithms in support of flow-based traffic classification. In this paper, we propose an AIS inspired algorithm for flow-based traffic classification, and evaluate its performance with and without the use of kernel functions. We utilize a publicly-available dataset to compare our results with other approaches that have been proposed in the recent literature. We provide several measures of the classification performance of the algorithm, as well as share our experience on the best features of the algorithm for this particular application. We also evaluate the proposed algorithm, comparing it with two other classification algorithms, and draw conclusions based on our findings. The algorithm generalizes well and gives high accuracy even with a small training set when compared to other algorithms, although the training and classification times were higher. The algorithm is also insensitive to the values of the input parameters, which makes it attractive for embedded and Internet of Things applications. The research presented here is a longer exposition of the work in [2].

Keywords*—artificial immune systems; internet traffic classification; multi-class classification; machine learning

I. INTRODUCTION

Because of problems with the fair use of the Internet, it has become more and more important in recent years to classify network traffic. There have been a few driving forces behind these developments, for example: (a) the use of the Internet to share copyrighted material, (b) the struggle between malicious hackers and security professionals, (c) the question of network neutrality [3].

Internet traffic classification has evolved over the years, and the techniques used to solve it have grown in complexity as well. Simply, classification of packet flows on a network can be accomplished by using well-known port numbers. This approach is fast and simple but is often inaccurate. The effectiveness of this approach to traffic classification has declined in recent years [4].

Another way to classify packet flows has been to inspect the data payload that a packet contains. This is often accurate, but is computationally intensive while also being easily fooled by encryption. To surpass the challenge of encrypted data, the host behavior based approach examines interactions between hosts can be compared to stored patterns. Another approach is to use the statistical features of a packet flow to classify it. This approach uses techniques from data mining and Machine Learning (ML). A characteristic of this approach is that it does not need to inspect the contents of packets to enable it to classify the flow, thus avoiding legal and ethical quandaries.

The focus of this paper will be to use the statistical features of a packet flow to classify it using a multi-class artificial immune system inspired algorithm. The remainder of the paper is organized as follows: in Section II, we survey some of the machine learning algorithms used in support of flow-based traffic classification. In Section III, artificial immune system classifiers are introduced, along with variations and improvements in support of multi-class classification. Section IV introduces our own AIS inspired algorithm. Sections V and VI describe our experimental setup and results. Finally, in Section VII, we draw conclusions based on our experimental results and discuss future research directions.

II. THE TRAFFIC CLASSIFICATION PROBLEM IN MACHINE LEARNING

The flow classification problem can be solved by using the statistical features of the information on the network. Some of the information that can be used to classify network flows is: port numbers, inter-packet delay, packet counts, as well as calculated features such as the averages and medians of these values.

In [5], Moore and Zuev applied a Naive Bayes classifier to the traffic classification problem. A simple naive Bayes classifier did not do well at first, with an average 65.3% classification accuracy. The accuracy rose, however, when kernel density estimation and Fact Correlation-Based Filtering (FCBF) were applied. The techniques were tested separately and jointly with the best performance achieved with all techniques used at the same time, achieving 96.3% average classification accuracy.

In [6], Alshammari and Zincir-Heywood, applied Support Vector Machines (SVM), Naive Bayes, RIPPER, and C4.5 algorithms using three publicly available datasets, focusing on classifying encrypted traffic. Singh and Agrawal [7] also applied several of the same ML algorithms as [6] to perform traffic classification, the algorithms being: Bayesian networks, multi-layer perceptron, C4.5 trees, Naive Bayes and the Radial Basis Function Neural Network.

To the best of our knowledge, our work makes the first attempt to apply an AIS inspired algorithm to the network traffic classification problem. Our work is motivated by the versatility of the immune system inspired algorithms and their low sensitivity to the values of the input parameters compared to other evolutionary, machine learning and optimization based approaches. Furthermore, AIS algorithms have been used extensively in Network Intrusion Detection systems, which also classify network traffic. [1]

III. Artificial Immune systems

In the following subsections, we provide a brief overview of Natural Immune Systems (NAS), training methods and their variations in support of multi-class classification.

A. *Natural Immune Systems*

NAS have developed to protect biological systems from outside threats, such as bacteria, viruses, and parasites, collectively known as pathogens. The role of NAS can be divided into two activities: recognition and elimination of pathogens. We focus on the recognition aspect of NAS in this paper, which differentiates between “self” (the tissues of the host), and “non-self” (everything else). The NAS can be divided into two sections: the innate immune system, which is fixed and not adaptable, and the acquired immune system, which adapts to the specific pathogens to which the host is exposed. The acquired immune system is of interest to computer scientists and has inspired a whole class of classification algorithms because of its ability to learn and recall from past experiences. The authors of [1] provide a good introduction to the use of AIS algorithms in support of intrusion detection systems.

B. *Training Methods*

The cells in the NAS that recognize pathogens and label them for disposal are called *B-cells* and *T-cells*. They are trained to recognize non-self tissues and pathogens in a two different ways, which are explained below.

Each B-cell generates detectors, which are called “antibodies”, that are able to recognize pathogens through the particular proteins shown on their surface. Through the process of negative selection, B-cells are generated randomly, and then destroyed in the thymus before they mature, if their antibodies match “self” tissues. In this way, only B-cells that recognize non-self tissues are kept in the body.

Clonal selection occurs when a B-cell or T-cell detects a pathogen, copying itself many times to start the immune response. During copying, the cells are subjected to errors which mutate the new cells slightly, and change the detector. Through clonal selection and the accompanying mutations, a population is able to adapt itself to an attack on the organism, and do so with the minimal amount of resources. The process has some similarity to genetic mutations. Also, clonal selection can be controlled through limited parameters and does not require the definition of potentially complex encoding, mutation, cross-over and selection operators as in the case of genetic algorithms.

Negative selection and clonal selection have been adapted into classification and optimization algorithms. An antibody within an AIS algorithm is an implementation of a simple classifier. Artificial immune systems used for classification are ensemble classifiers.

C. *Multi-class Classification Using AIS*

Although artificial immune system classifier algorithms naturally work as two-class classifiers ("self" and "non-self"), there has been work done to extend the techniques used for two-class classification to multi-class classification using AIS principles.

The earliest research that uses AIS for multi-class problems is Goodman, Boggess, and Watkins [8], in which the Artificial Immune Recognition System (AIRS) is proposed and tested against Kohonen's Learning Vector Quantization (LVQ). The concept of a resource-limited artificial immune system was introduced by Timmis and Neal in [9] and later [10]. Similar research on the subject of multi-class AIS algorithms was published by Cheng and Cheng [11]. In this publication, a hybrid algorithm combining AIS and SVM is proposed and evaluated. The application was the diagnosis of thyroid diseases. The classifier achieved 99.87% accuracy. Another publication that deals with multi-class AIS algorithms is [12] by Greensmith and Cayzer, which applies AIS to Internet document classification. In [13], Carter introduced several generalizations from previous research into an algorithm called Immunos.

In [14], Markowska-Kaczmar and Kordas applied the negative selection principle to train a multi-class AIS algorithm. Their algorithm develops a set of antibodies for every class in the dataset, and uses the negative selection principle. To classify a data point, each antibody is tested against the population, the class of antibodies that matches the data point the least number of times is the class assigned to the data point.

Negative selection is commonly used in AIS algorithms, however we employ a positive selection technique, which is essentially the reverse algorithm, which covers the space occupied by a class. Since negative selection proved to be a very slow training algorithm we propose a much faster training algorithm. We also develop an alternative method for multi-class classification. The details of our algorithm are provided in the following section.

IV. Algorithm Description

Our AIS inspired algorithm aims to classify packet flows into application classes. A packet flow is a sequence of packets from one host to another. A packet flow is defined by a 5-tuple consisting of source host, destination host, source port, destination port and transport protocol.

The dataset made available by Moore et al. [5] is used. The number of features used by our algorithm is reduced to 11, from the original set of 249 features, according to work done in the same publication. The feature reduction process is a very important step in machine learning, and it ensures that only the most useful features are used to make predictions, thus saving time and memory. The authors of [5] use a Fast Correlation-

Based Filter (FCBF) to perform feature reduction on the dataset. We chose not to duplicate their work for our algorithm and use only their reduced feature dataset to test our algorithm. The features are detailed in [5], and follow the recommendations of [15]. The features used are listed in Table I.

Even though the dataset is over 10 years old, we think it is a good fit for our work since we are able to compare our algorithm directly to the work published in [5], which is very useful. Our algorithm classifies each packet flow into one of 12 classes. The classes are listed in Table II, along with the applications that generated the traffic. The FTP application class listed in the table is separated into three different classes within the dataset, encompassing control, passive and data FTP flows, respectively.

Before our algorithm is applied to a network flow, the relevant information must be extracted from the traffic on the wire, in this section we assume that this is already done and we only describe the workings of the training and classification algorithms. The pseudo code of our immune system inspired algorithm is found in Figure 1. In the algorithm, each antibody has a location within the *n*-dimensional feature space (*n*=11 in our case), as well as a radius which denotes the boundary within which a data point will *match* the antibody. The training and testing data is assumed to be normalized. To initialize the population of antibodies, the training data is randomly sampled with replacement, and the radius is initialized to zero. Each class is allocated an equal portion of the antibody population.

The training of the population of antibodies happens as a one-shot algorithm. Every antibody present in the population matches one point in the training set at the beginning of the training phase, since the coordinates of each antibody are centered on one training point. After this, every antibody's radius is iteratively increased by a fixed step size, until the antibody matches a data point that is not of its designated class. Once the antibody reaches a non-self data point in the training set, it decreases its radius by the same step size, in order to prevent a misclassification. The algorithm uses a function called error_count, which returns the number of training points that the given antibody misclassifies.

TABLE I. CLASS LABELS AND APPLICATIONS[5]

Feature	Description
Port, *server*	Port Number at server
Number of pushed data packets, *server->client*	# of packets with the PUSH bit set in the TCP header
Initial window bytes, *client->server*	# of bytes in the initial window
Initial window bytes, *server->client*	# of bytes in the initial window
Average segment size, *server->client*	The average segment size
IP data bytes median, *client->server*	Median of total bytes in IP packets
Actual data packets, *client->server*	# of packets with at least a byte of TCP data payload
Data bytes in the wire variance, *server->client*	Variance of # of bytes in Ethernet packet
Minimum segment size, *client->server*	The minimum segment size
RTT samples, *client->server*	The total number of Round Trip Time (RTT) samples.
Pushed data packets, *client->server*	# of packets with the PUSH bit set in the TCP header

TABLE II. CLASS LABELS AND APPLICATIONS[5]

Class Label	Applications
FTP-CONTROL, FTP PASV, FTP-DATA	FTP
DATABASE	Postgres, Sqlnet, Oracle
INTERACTIVE	SSH,rlogin, telnet
MAIL	IMAP, POP2/3, SMTP
SERVICES	X11, DNS, ident, LDAP, NTP
WWW	WWW
P2P	KaZaA, BitTorrent
ATTACK	Worm and virus attacks
GAMES	Half-Life
MULTIMEDIA	Windows Media Player

During the classification phase, the captured features of the network traffic flow that needs to be classified is matched against the population of antibodies. The class of the first antibody that covers the captured features by its radius provides the classification decision. If no antibody is found to cover the captures features, the class of the closest antibody provides the classification decision. In such cases, the proposed algorithm works like the k-NN algorithm, when k is equal to 1. If the k-NN algorithm is unable to classify a pattern because it is the same distance from more than one antibody, then one of the classes that the antibodies belong to is chosen at random.

Throughout the tests done for this paper, the radius increase step size was chosen to be 0.01. If this value is too large, the classifier performance will suffer, if it is too small, the training will take longer. In general, this value should be chosen to be half the distance between the two closest points in the training dataset.

In this work, we worked exclusively with the Euclidean distance function to measure distances between points. We also experimented with the linear, polynomial and Gaussian kernels to try to increase the accuracy of the algorithm, as previously done in [16]. Kernel functions are used in other classifiers to deal with data that is not linearly separable into classes. Kernel functions are able to deal with this by projecting the data into higher dimensional spaces, were it can be more easily separated. Kernel functions are used extensively with SVM classifiers.

V. MODEL VALIDATION AND EXPERIMENTAL SETUP

Model validation was done using a 10-fold cross validation process. This means that the dataset used to train and test the algorithm is sampled from the original dataset and 10 sub datasets are created. Stratification is used as well, which means that the 10 subsets are sampled so that each class is evenly represented in each subset. The split between testing, validation, and training sets is 10%/10%/80%, respectively.

Even though the dataset contains 370,000 flows, we chose to limit the number of flows that we use to train and test the algorithm. The reason is that using larger datasets would cause the accuracy of the algorithm to unpredictably increase in some cases. This would give unreliable results and prevent us from evaluating the true performance of the algorithm.

Using 10 sub datasets, each test was performed 10 times, and at the end, the results were averaged. The tests are based on the work in [17] and [18], and follow the recommendations found in [3]. The algorithm is coded in Python. The algorithm

```
Definitions:
training_set: set of training vectors, class label is the first element of the vectors
classes: the set of class labels collected from the dataset
antibody: the basic element of the population
population: the set of antibodies used to classify flows
error_count: function which returns the number of misclassifications performed by an antibody
step_size: a parameter given to the algorithm
population_size: a parameter, the desired size of the antibody population
1. Initialization:
    training set = normalize(training_set)
    population = {}
    FOREACH { c | c ∈ classes }
        class_data = { i | i ∈ training_set, i[0] = c }
        counter = 0
        WHILE counter < $\left\lceil \frac{\text{population\_size}}{|\text{classes}|} \right\rceil$
          new_antibody=[center=random(class data), radius=0.0, class=c]
          population = population ∪ new_antibody
          counter = counter + 1
        ENDWHILE
    ENDFOREACH
2. Training:
    FOREACH {p | p ∈ population}
        changed = True
        WHILE changed
            IF error_count(p) > 0
                p[radius] = p[radius] - step_size
                changed = False
            ELSE
                p[radius] = p[radius] + step_size
                changed = True
        ENDWHILE
    ENDFOREACH
3. Classification:
    distances = {}
    FOREACH { p | p ∈ population }
        d = distance(pattern, p)
        IF d <= p[radius]:
            return p[class]
        ELSE
            distances = distances ∪ {(p[class], d)}
    ENDFOREACH
    a = argmin distance(pattern, a) {a | a ∈ population}
    return a[class]
```

Fig 1. Immune system inspired Network Flow Classification.

was tested four times, without a kernel, with a polynomial kernel, a linear kernel and a Gaussian kernel. The parameters for the kernels were chosen using a simple grid search technique with the validation set described above. The kernel functions were implemented as described in [16].

We also compared our algorithm to Naïve Bayes and SVM classifiers. Since all of the variables in the dataset are continuous, we used a Gaussian Naïve Bayes classifier, in which the model parameters were calculated using Maximum Likelihood Estimation. For the SVM classifier, we used a "one-versus-all" classifier without any kernels. We have chosen to display the results from these classifiers in the same figures in the interest of saving space.

VI. EXPERIMENTAL RESULTS

The first experiment performed was to calculate the classification accuracy vis-a-vis the size of the population of antibodies, as seen in Figure 2. The size of the dataset was held constant at 1000 flows, and the proportions of the testing, validation and training sets were 10%/10%/80% respectively, and is the same for every test in this paper. The size of the population of antibodies was increased from 200 to 1000 members. The accuracy of the algorithm rose from 81.8% to 91% without the kernel trick. The best performance on this test was achieved with the linear kernel, with 92.3% accuracy. This figure also includes the accuracy of Naïve Bayes and SVM algorithms, which were trained with datasets of the same size as the AIS algorithm. These two classifiers do not use an antibody population to perform classification, and are included to serve as a baseline of comparison. The maximum accuracy of the Naïve Bayes algorithm is 82.2%, and the maximum accuracy of the SVM classifier is 44.1%.

The second experiment performed was used to study the classification accuracy achieved by the algorithm as the size of the dataset is increased. In this experiment, the size of the dataset was increased from 200 to 1000 elements. The results are shown in Figure 3. The antibody population was held constant at 1000 members. The accuracy of the algorithm remained fairly constant, varying between 86% and 92.3% without the kernel trick. The best performance was achieved by the polynomial kernel, with 93.6% accuracy. The accuracy of Naïve Bayes and SVM classifiers is also shown. The SVM classifier achieved a maximum accuracy of 42.6%, and the Naïve Bayes classifier achieved a maximum accuracy of 83.5%.

Figure 3 illustrates the ability of the algorithm to generalize well from small samples of data, especially when compared to the SVM and Naive Bayes algorithms. The highest accuracy was achieved between 200 and 500 training data points, and decreases after that.

Kernels improve an algorithms' performance by projecting the data into a higher-dimensional space in which the class boundaries are more easily defined. Figures 2 and 3 illustrate that using kernels does not improve the performance of the algorithm significantly. This observation is particularly important when utilizing our classification problem in memory constrained embedded systems, as in the case of the Internet of Things (IoT).

The third experiment performed was used to calculate the amount of time required to classify 100 elements in the testing set. The value of 100 elements is a side effect of the fact that the size of the dataset was held constant at 1000, with a 10%/10%/80% split between testing, validation, and training sets, respectively. The antibody population was increased from 200 elements to 1000 elements, and the algorithm was trained with a dataset size held constant at 1000 elements. The results are illustrated in Figure 4, with the time displayed in seconds. The classification time is linearly related to the number of antibodies in the population. Naïve Bayes and SVM classifiers were trained on datasets of 1000 elements as well, and are included to serve as a baseline of comparison. These classifiers do not use a population of antibodies to perform classification, and are faster than our AIS algorithm, their lines run along the bottom of the graph.

The fourth experiment performed was used to calculate the amount of time required to train the population of antibodies against the size of the dataset. The size of the dataset was increased from 200 elements to 1000 elements, with the training data being 80% of that. The antibody population size remained constant at 1000 members. The results are shown in Figure 5, with the time displayed in seconds. It can be seen that the relationship is roughly linear. The SVM and Naïve Bayes classifiers were also trained with a dataset of the same size, their training times were faster and can be seen along the bottom of the graph. Even though the time required by our algorithm with and without kernel methods was largely the same, we chose to include the data in the charts for completeness.

When compared against the results from [3], one fact stands out: the AIS algorithm was able to achieve better or equal accuracy than all of the other classifiers with about 1/3 of the training samples. Specifically, the average accuracy of our AIS algorithm is about 89% when the training set contains about 300 samples, which is about equal to the performance of the best classifier tested in [3], the SVM classifier, when given the same training set size. Although our classifier does not exceed the classification accuracy of the best classifiers, it is very good at generalizing from small training sets. This conclusion is also supported by Figures 2 and 3.

To be able to assess the performance of the classifier on a per-class basis, the F-measure was used. The F-measure can be interpreted as the weighted average of precision and recall, and has a range between 0 and 1. For this experiment, the antibody population size was held constant at 1000, and the dataset size was increased from 200 to 1000. The F-Measure of a few representative classes are graphed in Figure 6. In order to display the “FTP” class in Figures 6 and 7, we took the average of the “FTP-CONTROL”, “FTP-PASV”, and “FTP-DATA” classes for the measurements graphed. When inspecting the F- measures achieved by our AIS, it can be seen that the AIS algorithm is able to get high F-measures across almost all classes, including classes with few training samples present in the dataset. This is especially visible with the “P2P” and “FTP” classes.

The precision and recall of the classifier were also calculated without using kernel functions. For this experiment, the antibody population size was held constant at 1000, and the dataset size was increased from 200 to 1000. The precision and recall of a few representative classes are graphed in Figure 7. All tests were performed on an Intel Core i5 running at 1.8 GHz with 4 GB of memory.

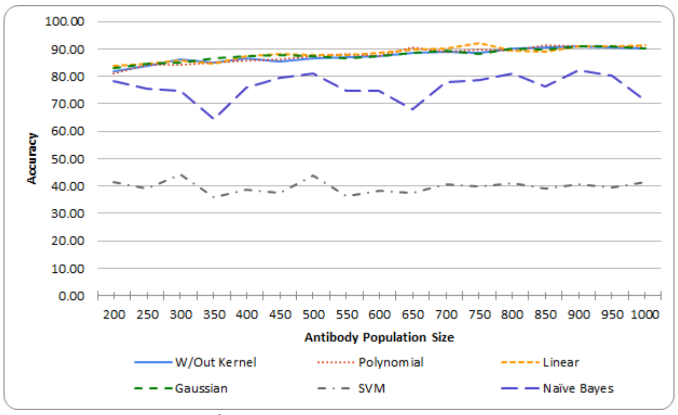

Fig 2. Classification accuracy and antibody population

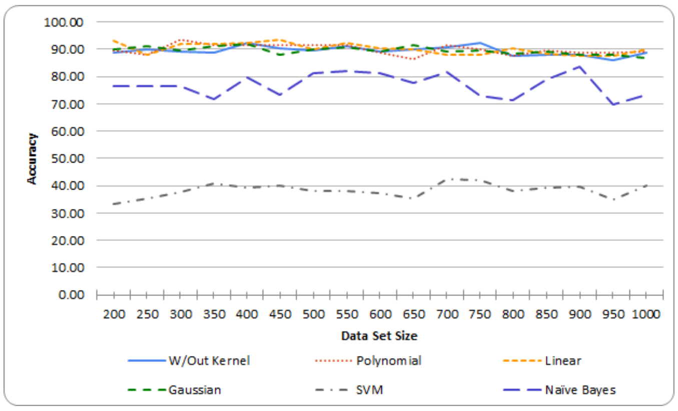

Fig 3. Classification accuracy and dataset size

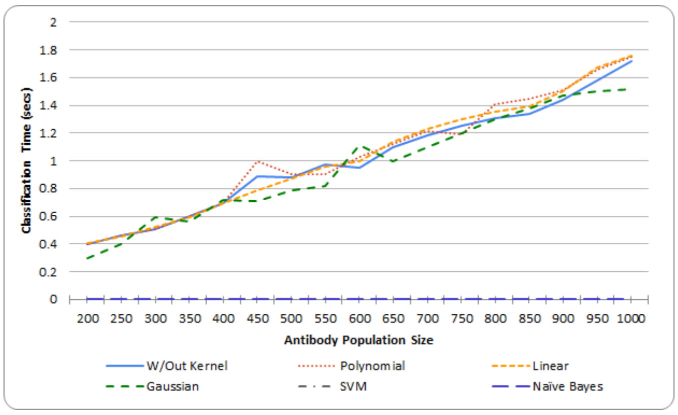

Fig 4. Classification time and antibody population size

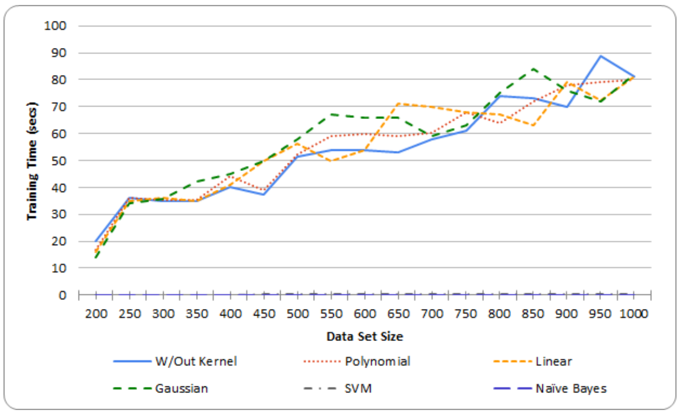

Fig 5. Training time and dataset size

VII. Conclusions and Future Work

In this paper we demonstrated the application of an immune system inspired classification algorithm to classify network flows according to application. We illustrated the performance of the algorithm in terms of accuracy, classification time, training time and F-measure. We also applied kernel functions to the algorithm and compare its performance to the Naïve Bayes and SVM algorithms. Although the maximum accuracy was achieved with a kernel function, we believe that the difference is not statistically significant.

The algorithm's accuracy is similar to other machine learning techniques used on this problem [17][5][7]. The algorithm is especially suitable for embedded applications because of its insensitivity to the use of kernel functions, as well as well as the few parameters required for its use. Since the algorithm does not need kernel functions to perform well, it does not requires parameters for kernel functions to be set. We were able to improve on the accuracy of Naïve Bayes and SVM classifiers when used with small training sets, but the training and classification steps of our AIS algorithm are slower.

In the future, the training and classification time of the algorithm could be improved with the use of a *k*-d tree or Bloom filter data structure. Furthermore, the algorithm is inherently data parallel and can benefit from GPUs to speed up the training and classification times.

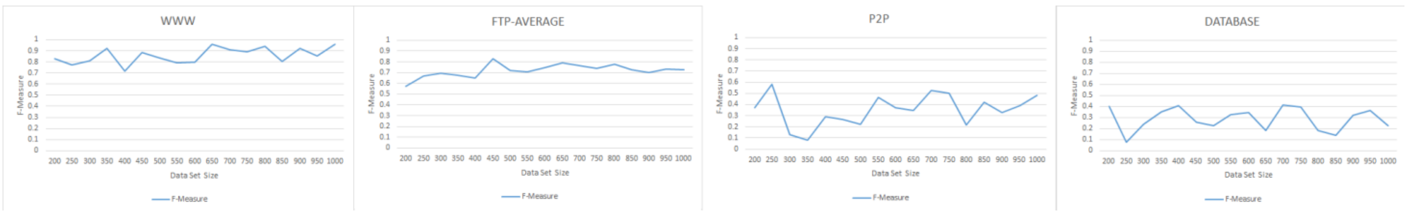

Fig 6. F-measure and Dataset Size

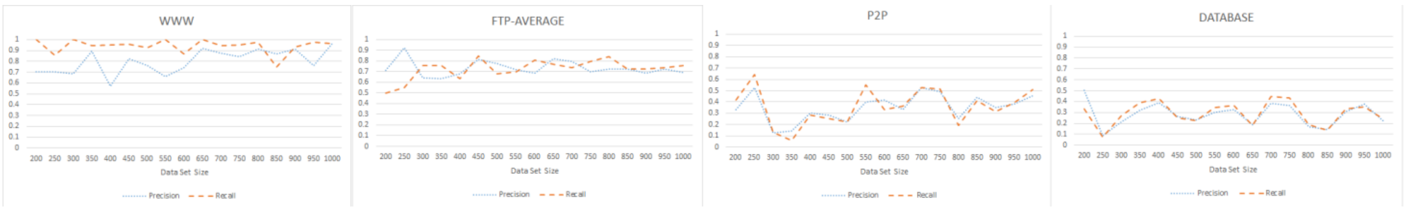

Fig 7. Precision, Recall and Dataset Size

References

[1] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco and J. Twycross, J. "Immune system approaches to intrusion detection–a review," in *Natural Computing*, pp. 413-466, 2007

[2] B. Schmidt, D. Kountanis, A. A;-Fuqaha, "Artificial Immune System Inspired Algorithm for Flow-Based Internet Traffic Classification," in *IEEE CloudCom 2014*, Big Data Track, Singapore, 15-18 December 2014.

[3] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*, pp. 11, December 2008

[4] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement*, pp. 41-54, 2005

[5] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33, No. 1, pp. 50-60, June 2005

[6] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA2009)*, pp. 1-8, 2009

[7] K. Singh and S. Agrawal, "Comparative analysis of five machine learning algorithms for IP traffic classification," in *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pp. 33-38, April 2011

[8] D. E. Goodman, L. Boggess, and A. Watkins, "Artificial immune system classification of multiple-class problems," in *Proceedings of the artificial neural networks in engineering*, vol. 2, pp. 179-183, 2002

[9] J. Timmis and M. Neal, "Investigating the evolution and stability of a resource limited artificial immune system," in *Proceedings of the genetic and evolutionary computation conference (GECCO)*, pp. 40-41, 2000

[10] J. Timmis and M. Neal, "A resource limited artificial immune system for data analysis," in *Knowledge-Based Systems*, 14(3), pp. 121-130, 2001

[11] H. P. Cheng, and C. S. Cheng, "A hybrid multiclass classifier based on artificial immune algorithm and support vector machine," in *3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMiA)*, pp. 46-50, 2011

[12] J. Greensmith and S. Cayzer, "An artificial immune system approach to semantic document classification," in *Artificial Immune Systems*, pp. 136-146, 2003

[13] J. H. Carter, "The immune system as a model for pattern recognition and classification," in *Journal of the American Medical Informatics Association*, 7(1), pp. 28-41, 2000

[14] U. Markowska-Kaczmar and B. Kordas, "Multi-class iteratively refined negative selection classifier," in *Applied Soft Computing*, pp. 972-984, 2008

[15] Y. S. Lim, H. C. Kim, J. Jeong, C. K. Kim, T. T. Kwon and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," in *Proceedings of the 6th International Conference*, pp. 9, November 2010

[16] T. S. Guzella, T. A. Mota-Santos &W. M. Caminhas, "Artificial immune systems and kernel methods," in *Artificial Immune Systems*, pp. 303-315, 2008

[17] T. T. Nguyen, and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," in *Communications Surveys & Tutorials*, 10(4), pp. 56-76, 2008

[18] S. Ubik and P. Zejdl, "Evaluating application-layer classification using a Machine Learning technique over different high speed networks," in *Fifth International Conference on Systems and Networks Communications (ICSNC)*, pp. 387-391, August 2011