Summer 2020

ANTA: Accelerated Network Traffic Analytics.

Matthew Grohotolski

Connor DiLeo

ANTA: Accelerated Network Traffic Analytics

Matthew Grohotolski*, Connor DiLeo*, Peilong Li*, Onur Barut†, Yan Luo†, Tong Zhang‡, Weigang Li‡
*Elizabethtown College
{*GrohotolskiM*, *DiLeoC*, *LiP*}@etown.edu
†University of Massachusetts Lowell

Onur_Barut@student.uml.edu, *Yan_Luo*@uml.edu
‡Intel Corporation
{*Tong*2.*Zhang*, *Weigang.Li*}@intel.com

***Abstract*—Implementing traditional machine learning models and neural networks has become trivial in detecting malicious network traffic and has sparked interest in many researchers investigating this field. Standard implementations include using the baseline models in packages such as sklearn, tensorflow, and keras. In this paper we seek to advance the field of network detection and produce results which will have great benefits in terms of speed and performance of these models. We take advantage of Intel's DAAL and OpenVINO packages as they are the two best performance enhancing methods which are publicly available today. Furthermore, comparisons will be made to determine the impact of these two Intel packages on network intrusion detection.**

***Index Terms*—Encrypted Traffic Analysis; DAAL; OpenVINO; Edge Computing; uCPE;**

I. Introduction and Motivation

Since the dawn of the Internet, there have been challenges with malicious users manipulating network traffic and causing trouble for Internet users and businesses. These challenges have only gotten worse as the Internet has progressed, and businesses are constantly evolving their network infrastructure. With the implementation of encryption on websites, manipulated data can be very hard to track accurately. According to Google, 97% of all websites are encrypted [1] as of July 2020. The number of daily issued certificates keeps increasing linearly and is currently sitting at 1.5 million [2]. With this knowledge, it is more important than ever to prevent further attacks from occurring on encrypted traffic.

Although this problem seems complex in nature, many researches have advanced their methods for detection of malicious network traffic in order to prevent further attacks. Previous attempts to detect such traffic have been successful, but oftentimes take too much time for detection and are impractical for analyzing the large influx of data which has been made readily available. Alongside these complications, many researchers tend to use different datasets when comparing their newly created models. This confusion leads to different results when training and testing models. In the past, it has been difficult for researches to find appropriate, and complete datasets; This is where the common ground of network traffic analysis comes in.

Within this project we use the NetML and CICIDS2017 datasets which are publicly available and widely used by the network detection community to compare the results of their models. We also work toward improving previous detection rates by implementing Intel's DAAL and OpenVino to both Machine Learning and Deep Learning models. Our goals that we wish to accomplish are as follows. 1) We first implemented traditional Machine Learning and Deep Learning Models in order to get baseline scores 2) We then seek to accelerate the traditional Machine Learning and Deep Learning models with the use of Intel's DAAL and OpenVino. 3) Lastly, we conduct a thorough evaluation of multiple variables in order to determine improvement.

The organization of this paper is as follows. Section §II talks about all the previous works that have been accomplished by previous researchers. Section §III depicts the overall design of OpenVino and DAAL. Section §IV we talk about performance evaluation. Lastly, in Section §V we conclude the paper.

II. Related Works

Anderson et al. [3], [4] implemented a group of machine learning algorithms for ETA. They felt that there were two reasons as to why traditional matching could not be applied to ETA, those being inaccurate ground truth and a highly non-stationary data distribution. What the authors did in order to overcome these problems was that they developed machine learning models that had a unique and diverse set of network flow data features. Lastly, there is the work done with the `ACETA` research. Their work consisted of the use of Joy in the extraction stage of the network traffic data. Joy is an open-sourced software that was created by the same authors Anderson et al. [3], [4]. They also implemented accelerated versions of their models with the use of Intel's DAAL and OpenVino. Our research with `ANTA` continues the work that was previously done with the `ACETA` research.

III. Design

In this section, we present the design of our project by firstly talking about the vanilla implementation of our models. Then we talk about accelerating those models with the use of Intel's DAAL and OpenVino.

Firstly, we have our machine learning models which were important from the scikit-learn library. In this project we tested a total of four models that would be classified under machine learning. Those models are as follows: Logistic Regression(LR), Recurrent Neural Network(RNN), Support Vector Machine(SVM), and lastly k-Nearest Neighbors(kNN).The workflow of DAAL's workflow can be seen in figure 2 Before the models can be used for testing the datasets, the data must first be processed. The process can be seen in the figure 1

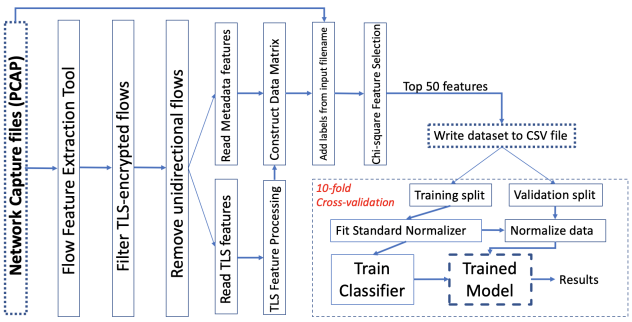

Fig. 1. Dataset Process

Fig. 2. Intel's DAAL Workflow

As previously mentioned, the machine learning models are not the only models that we tested. There were also a total of four deep learning models that were used. Those models are as follows: Artificial neural network (ANN), Convolutional neural network (CNN), Long short-term memory(LSTM), 1D Convolutional Neural Network(1DCNN), and lastly 2D Convolutional Neural Network(2DCNN). Just like in the machine learning section, the data is processed the same way in order to prepare it.

Once we tested and competed our vanilla models in order to get the baseline results, it was time to accelerate them. For the machine learning models we used Intel's Data Analytics Acceleration Library(DAAL). DAAL is a library that can be used by developers in order to accelerate data analytics. The advantages that DAAL has over other libraries is that DAAL optimizes the entire workflow. What this does is, it ensures that the entire process is accelerated from data acquisition, to training and predictions.

Deep learning is a field of machine learning which has been evolving since its creation. We leverage standard neural network models found in tensorflow and keras to implement malware detection with deep learning. Various forms of these neural networks were implemented which include: Artificial Neural Network (ANN), 1D Convolutional Neural Network (1D CNN), 2D Convolutional Neural Network (2D CNN), Long Short Term Memory Neural Network (LSTM), and lastly Convolutional Long Short Term Memory Neural Network (CNN+LSTM).

Neural networks are more complex than traditional machine learning algorithms because they consist of hyper parameters which can be fine tuned to yield better results. To obtain the optimal hyper parameters for each model, we first tested a multitude of values for each variable and use a combination of the best yielded results to use with the final implementation of each neural network model. A list of all optimized variable names is shown in in table I.

TABLE I
Deep Learning Hyper-Parameters

Feature
learning_rate
decay_rate
dropout_rate
n_batch
n_epoch
filters
kernel_size
strides
CNN_layers
clf_reg

Some of the deep learning models resulted in slower than expected inference times. In order to solve this we used Intel's OpenVINO library to accelerate our baseline neural networks. OpenVINO is built upon two main components, its Model Optimizer and Inference Engine. The Model Optimizer is responsible for taking pre-trained models from our standard neural network models such as ANN, CNN, and LSTM; it then transforms these models into a format which OpenVINO can utilize. After the model is transformed, we utilize OpenVINO's Inference Engine to drastically decrease inference times and are able to see a huge performance boost when compared to the standard models which were previously converted. The full process for OpenVINO can be found below in figure 3

Once all DAAL and OpenVINO models were fully implemented, we put them to the test against their standard counterparts in the next section and compare statistics collected from each model.

IV. Evaluation

In this section of the paper we are evaluating the important aspects that are proposed `ANTA`. Specifically, we

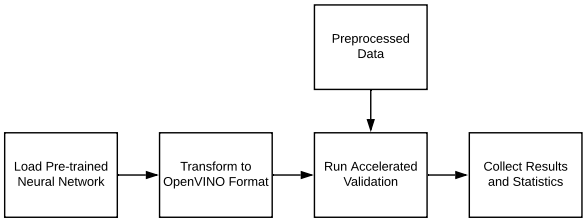

Fig. 3. Intel's OpenVINO Workflow

will firstly talk about the datasets used throughout this research. Then we will talk about the specs used to train and test these models. Lastly, we talk about the general improvements that were made with the implementation of Intel's OpenVino and DAAL.

A. Datasets

All of the testing done throughout `ANTA` was done with two datasets NetML and CICIDS2017. These dataseta were provided to us by the people at the University of Massachusetts Lowell who took the original versions of these datasets, and cleaned them up so it would be easier to test on. The NetML dataset was created with TLS encrypted malware detection and consists of 114k encrypted flow samples. It also consisted of 20 different types of malware and benign classes. The CICIDS2017 dataset was also created with TLS encrypted malware detection and consists of 75k encrypted flow samples consisting of infiltration malware and benign classes.

B. System Specifications

Throughout this research our data was being collected with a computer who's specs can be seen in Table II. We used built in Python tools in order to collect our data, like training time, inference time, accuracy, and CPU usage.

TABLE II
Experiment Platform Specification

Item	Specification
CPU	Intel i7 8700k 6 cores 12 threads @ 3.7GHz
GPU	GTX 1080ti with 11Gb DDR5
Memory	16 Gb DDR4 @ 3200MHz
HDD	4Tb HD @ 7200RPM
Software	Intel DAAL v2020.1
	Intel OpenVINO v2020.3
	Anaconda v5.3.0
Host OS	Ubuntu 18.04 Desktop

C. Performance Results

The user is given many options towards how they want to test the models. The way the models are currently implemented allows customization and automation. Firstly, the user will need to pick the models that they will want to test. The current options, as well as their accelerated version, are as follows: Decision Forest, kNN Logistic Regression, SVM, 1D CNN, 2D CNN, LSTM, and a combination of CNN and LSTM. The user can pick to run as many of these models as they desire in one go. After the user declared what models he wanted to test he can then choose between the two datasets that we have implemented, which are NetML and CICIDS2017. Lastly, there is the amount of times the user would like to test the models. After all of these customization features has been determined, the program is then ran. While the models are being tested, a handful of data is also being collected. The data that is being collected is the models accuracy, max CPU levels, false alarm rate (FAR), true positive rate (TPR), preprocessing time, and lastly training time. Since the user has the option to run the models multiple times, not only will each iteration have its numbers written down but a mean for all the values will automatically be calculated. It is encouraged to have the models run multiple times in order to get a solid average of the data. After the models have been trained and tested it is then time to evaluate the data.

D. Training Time

Figure 4 and 5 both represent the average training time that took place for each machine learning model, as well as their accelerated version. As we can see, the DAAL version of of each model generally either stayed roughly the same. The only major out liar that will consistently appear throughout our results are the Linear Regression models, as the vanilla model seems to outperform in both accuracy percentage and training time.

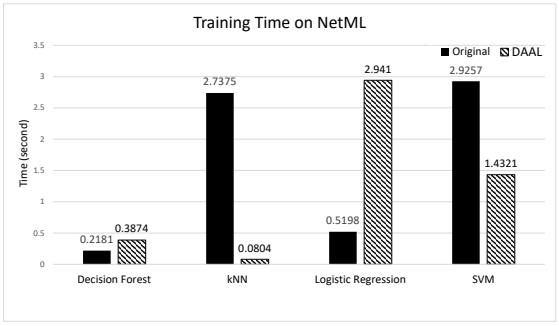

Fig. 4. DAAL Training Time on NetML

Just like the previous figures, figures 6 and 7 both represent the average training time but this time for the deep learning models.

E. Accuracy Percentage

Another key component that we are looking for while we accelerated our models was the accuracy. It is important that the accuracy stays close to the original once we

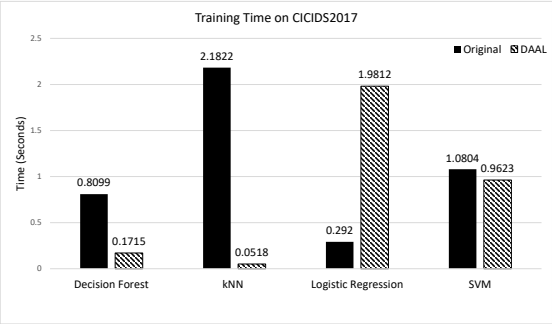

Fig. 5. DAAL Training Time on CICIDS2017

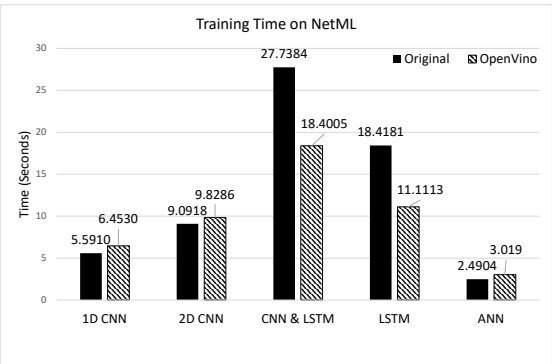

Fig. 6. OpenVino Training Time on NetML

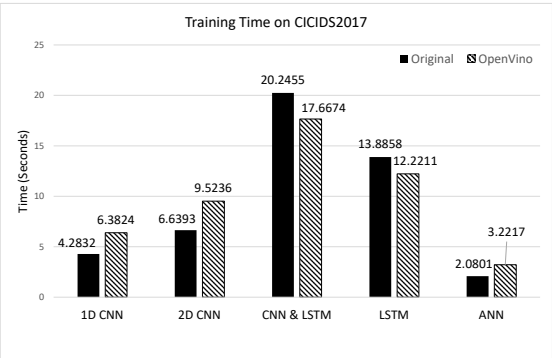

Fig. 7. OpenVino Training Time on CICIDS2017

accelerate the models. Figures 8 and 9 both show the accuracy with all of the machine learning models, as figures 10 and 11 show the accuracy with the deep learning models.

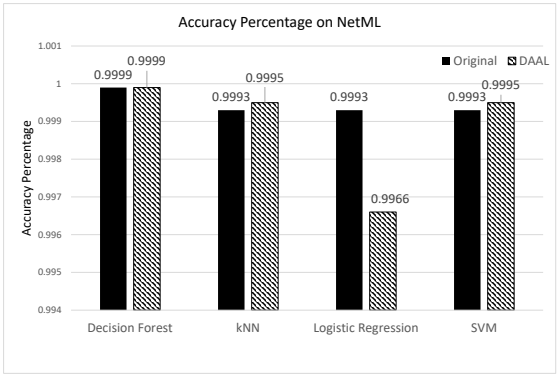

Fig. 8. DAAL Accuracy Percentage on NetML

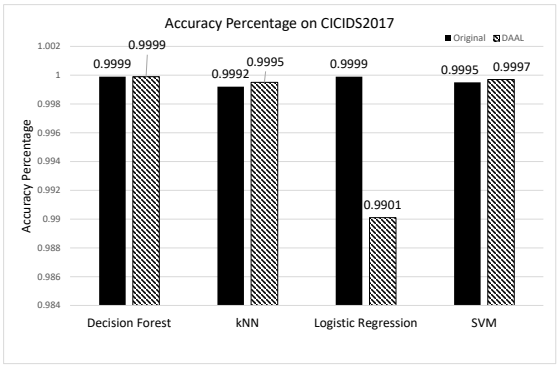

Fig. 9. DAAL Accuracy Percentage on CICIDS2017

Coming back to hyper parameter fine tuning of our neural networks we saw improvements around the board on all models from testing and tuning our parameters for each model. Each model was ran with different variables for each hyper parameter, which we used to collect the best parameters for each model. Once all the variables were collected we plugged them into our neural networks and ran the models 10 times to obtain average results for each model. Lastly, a comparison was made to the original model accuracy values to see how much of a difference hyper parameter tuning makes with deep learning. The graph comparing these two modes of testing are shown in Figure 12. The models which benifited the most from the hyper parameter implementation were the 1D CNN,

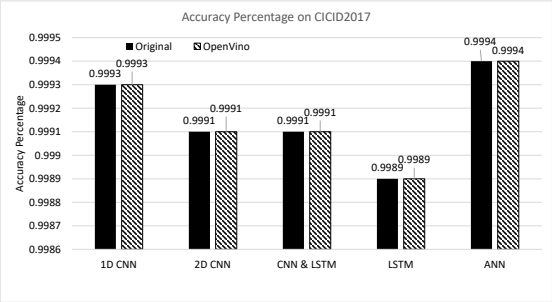

Fig. 10. OpenVino Accuracy Percentage on CICIDS2017

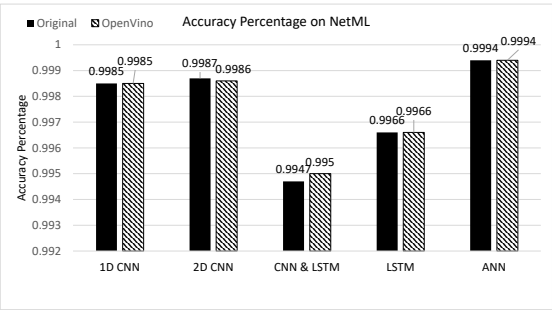

Fig. 11. OpenVino Accuracy Percentage on NetML

LSTM, and CNN+LSTM; each of these models improved an average of .01%.

V. Conclusion

Within this paper, we strive to accelerate past machine learning and deep learning models so that they can be used in real-world scenarios. This is done using publicly available datasets, namely NetML and CICIDS2017, which contain over one million network flows of information regarding malicious and benign network traffic. After performing prepossessing on the data, we accelerate previous models for detection in order to significantly reduce the time it takes for inference results to detect malicious data from future collected network flows. Eventually, this process could be automated to include live-fed processed data into these models in order to create live detection over the network.

We have made the source code for this project open-source for future research and is available to access through GitHub: https://github.com/BlueJayADAL/SCARP2020-ML.

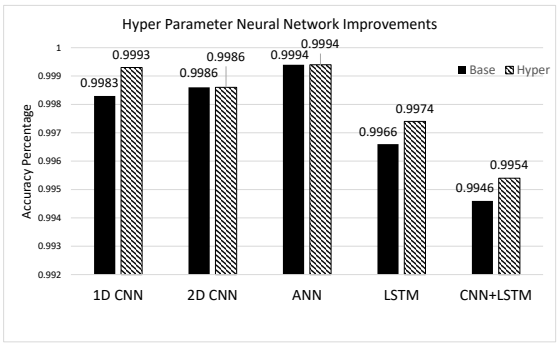

Fig. 12. Hyper Parameter Neural Network Improvements

Acknowledgment

This work is supported in part by the National Science Foundation (No. 1547428, No. 1541434, No. 1738965 and No. 1450996), a grant from Intel Corporation, and a grant from Summer Scholarship, Creative Arts and Research Projects (SCARP) Program of Elizabethtown College.

References

[1] Google, LLC, "Https encryption on the web," 2020. [Online]. Available: https://transparencyreport.google.com/https/overview

[2] Let's Encrypt, "Let's encrypt stats," 2019. [Online]. Available: https://letsencrypt.org/stats/

[3] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, ser. AISec '16. New York, NY, USA: ACM, 2016, pp. 35–46. [Online]. Available: http://doi.acm.org/10.1145/2996758.2996768

[4] ——, "Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 1723–1732. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098163