A Summary of the TAUTeam Approach to Wining in the NetML Challenge 2020

Eyal Horowicz*, Tal Shapira† and Yuval Shavitt‡

School of Electrical Engineering, Tel-Aviv University
Email: *eyalhorowicz@mail.tau.ac.il, †talshapira1@mail.tau.ac.il
‡shavitt@eng.tau.ac.il

I. Introduction

NetML Challenge 2020 [1] is a machine learning driven network traffic analytic challenge where participants are required to apply novel machine learning technologies in order to detect malicious flows or distinguish between applications in a fine-grained fashion among network flows.

In the challenge's dataset, each flow is represented by 121 features extracted from up to 200 packets in both directions as described by Barut *et al.* [1] and is labeled by 2 or 3 hierarchical multi-labels. The dataset consists of records of over 1M flows and relies on 3 public sources: NetML [2], CICIDS2017 [3] and non-vpn2016 [4].

The challenge is composed of 7 tracks, 4 of them for malware detection and the rest are for traffic classification:

- Track 1 – Malware Detection using top-level annotations with NetML dataset
- Track 2 – Malware Classification using fine-grained annotations with NetML dataset
- Track 3 – Malware Detection using top-level annotations with CICIDS2017 dataset
- Track 4 – Malware Classification using fine-grained annotations with CICIDS2017 dataset
- Track 5 – Traffic Type Classification using top-level annotations with non-vpn2016 dataset
- Track 6 – Application Classification using mid-level annotations with non-vpn2016 dataset
- Track 7 – Fine-grained Application Classification with non-vpn2016 dataset

In order to win a track, team must achieve the highest score. There are 2 types of scores:

- Multi-Class classification score:
  $(F1\_Score) \cdot (Mean\_Average\_Precision)$
- Binary classifications score:
  $(True\_Positive\_Rate) \cdot (1 - (False\_Alarm\_Rate))$

The TAU-Team is the only team to capture one of the top 3 places in all of the tracks and first place in more than one. In this document, we share technical details of our approach for Track-3 and Track-5, where we got first place.

II. Methods

In this section, we describe how we pre-processed the data by removing most of the features and creating a few new features, the specific classifiers we tried and the strategies we used for the training and predicting process.

Table I: Original features of NetML challenge

Feature	**Description**
time_length	period time of the sample
pr	protocol (6 or 17)
src_port	source port
dst_port	destination port
bytes_out	total bytes out
num_pkts_out	total packets out
bytes_in	total bytes in
num_pkts_in	total packets in
intervals_ccnt[]	histogram of packet arrive intervals
ack_psh_rst_syn_fin_cnt[]	histogram of tcp flag counting
hdr_distinct	#distinct values of header len
hdr_ccnt[]	histogram of header len
pld_distinct	#distinct values of payload length
pld_ccnt[]	histogram of payload len
hdr_mean	mean value of header len
hdr_bin_40	#packets with header len $\in [28\,40]$
pld_bin_128	#packets with payload len < 128
pld_bin_inf	#packets with payload len > 1024
pld_max	max value of payload length
pld_mean	mean value of payload length
pld_medium	medium value of payload length
pld_var	variance value of payload length
rev...	flow features of the reverse flow

A. Data pre-porcessing

The supplied datasets consist of a unique representation of bi-directional flows. Each flow is annotated by (as it termed by [1]) meta-data features, but since some of these are histograms, and there are additional reverse features, there are 121 features in total. The full description of the original meta-data features is shown in Table I.

We created new features by calculating rates, mean sizes of packets and mean of times between packets in both directions of the flow. We also added features for the ratios between both directions. See Table II for full description. We examined different combinations of the original features and found that many classifiers achieved almost the same or even better

Table II: New features

Feature	Description
bytes_in_out_ratio	in / out of bytes
num_pkts_in_ratio	in / out of num_pkts
bytes_in_rate	bytes_in / time_length
bytes_out_rate	bytes_out / time_length
num_pkts_in_rate	num_pkts_in / time_length
num_pkts_out_rate	num_pkts_out / time_length
avg_pkt_in_bytes	bytes_in / num_pkts_in
avg_pkt_out_bytes	bytes_out / num_pkts_out
avg_dt_in	time_length / num_pkts_in
avg_dt_out	time_length / num_pkts_out
avg_pkt_in_out_bytes_ratio	in / out of avg_pkt_bytes
avg_dt_in_out_ratio	in / out of avg_dt

results without the 68 histogram features of header length, payload length and packet inter-arrival intervals.

We normalized all features to have the same mean and variance. For the normalization we used all the data, and not limited the process to the labeled data.

B. *Classifiers*

In classification problems with small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class. Thus, we experimented with several of those and found Random Forest [5] and XGBoost [6] to be the most powerful classifiers.

Another, new promising approach was to use deep learning classifier with a sequential attention mechanism, which allows it to choose the features to reason from at each decision step (We used Tabnet [7]). However, the deep learning classifiers achieved lower results than the decision-tree-based classifiers, and its execution time was significantly longer.

C. *Training Specification and Strategies*

1) One Vs. Rest: Also known as One Vs. All, this strategy consists of fitting one classifier per class. For each classifier, the class is fitted against all the other classes.

2) k-fold cross-validation: In k-fold cross-validation, the original sample is randomly partitioned into k equal sized sub-samples. Of the k sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining k-1 sub-samples are used as training data. The cross-validation process is repeated k times, with each of the k sub-samples used exactly once as the validation data. We used k-fold with k=5.

3) Fine Labeled: In a hierarchical classification task, where each example simultaneously belongs to one class in each hierarchical level, several strategies can be applied in order to utilize the hierarchical structure information. On the top level we have used the Fine Labeled and the One Vs. Rest strategy together, creating multiple classifiers for each top level label and using them all in order to deduce the best tag.

III. Experiments and Results

The annotations for the test-std and test-challenge of NetML were not released, therefore we report here results from experiments using the development dataset, as was done in the public baseline.

A. *Features selection*

We trained models with several sets of features: the original features (121), without the histogram features (53), without the histogram but with our new feature (65), and with all original + our new features (133). Table II shows that the differences between the feature sets are small: from almost none to 4%. One important conclusion from this table is that removing the histogram features that are more than half of all the features, have no real influence on the results, but it reduces RAM consumption and improves executing time, significantly. Thus, in Table IV and Table V we used the set with our new features but without the histogram features.

Table III: Overall score based on different features

	Original	Hist Removed	Hist Removed + New	Original + New
CICIDS Top RandomForest	0.9785	0.9790	0.9825	**0.9828**
CICIDS Top XGBOOST	0.9781	**0.9789**	0.9786	0.9782
NonVpn Top RandomForest	**0.2878**	0.2864	0.2854	0.2862
NonVpn Top XGBoost	0.1942	0.1945	0.1980	**0.2019**

B. *One Vs. Rest strategy*

The One vs. rest strategy is relevant only to multi class problems, and meaningless for the binary classifications. Therefore we present the results of the fine-level classification in Table IV. We found that this simple strategy was useful for the traffic classification task.

Table IV: Overall score, One Vs. Rest strategy

	Random Forest	Random Forest One-Vs-Rest	XGBoost	XGBoost One-Vs-Rest
CICIDS Fine	0.9089	0.9136	**0.9277**	0.9178
NonVpn Fine	0.0929	**0.0941**	0.0644	0.0687

C. *Fine Labeled*

We trained the top classifiers to predict the fine labels and aggregate to the top level based on fine prediction. The hierarchical structure of this task made this process extremely valuable, as can be seen by comparing the results in Table V with the ones in Table III - Using this strategy improved the Malware Detection overall score from 0.983 to 0.994 and improved the Traffic Classification score dramatically by 75%.

Table V: Overall score, Fine labeled strategy

	Random Forest	**Random Forest One-Vs-Rest**	**XGBoost**	**XGBoost One-Vs-Rest**
CICIDS Top	0.9932	**0.9937**	0.9886	0.9881
NonVpn Top	0.4982	**0.5027**	0.3878	0.3906

IV. Concluding Remarks

We described our approach for achieving the highest score in some of the traffic classification and malware detection tasks on NetML challenge. We revealed that histogram based features, which are more than half of the original features can be removed without damaging the classification. We also found that the hierarchical structure of the problems can be used to improved the classification task and showed a specific method combining the Fine Labeled and the One Vs. Rest strategies to achieve the highest score for CICIDS Top and NonVpn Top and reach the 1st place in tracks 3, 5.

References

[1] O. Barut, Y. Luo, T. Zhang, W. Li, and P. Li, "NetML: A challenge for network traffic analytics," 2020.

[2] Stratosphere, "Stratosphere laboratory datasets," 2015, retrieved March 13, 2020, from https://www.stratosphereips.org/datasets-overview.

[3] I. Sharafaldin., A. H. Lashkari., and A. A. Ghorbani., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *The 4th International Conference on Information Systems Security and Privacy (ICISSP)*. INSTICC, Jan. 2018, pp. 108–116.

[4] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *The 2nd International Conference on Information Systems Security and Privacy (ICISSP)*. INSTICC, Feb. 2016, pp. 407–414.

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A%3A1010933404324

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.

[7] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," 2020.