Identifying Malicious Traffic on Network-Data Streams using Convolutional Neural Networks and Focal Loss

1st Victor Peñaloza
RLICT: Research Laboratory in Information and Communication Technologies
Universidad Galileo
Ciudad de Guatemala, Guatemala
victorsergio@galileo.edu

***Abstract*—This paper presents a depiction of our participation in the NetML Network Traffic Analytics Challenge 2020 [1] from the NETAML Workshop 2020. The objective of this task is to analyze network data streams to ascertain malicious traffic on a top-level fashion and a fine-grained fashion. For this task, we proposed a classifier architecture based on a Convolutional Neural Network [2] to extract major features from data streams generated by malicious and benign software. For this, we proposed to convert the numerical data from the NetML dataset to a grayscale bitmap structure to feed a 2D Convolutional Neural Network and addressing the problem as an image-processing task. Additionally, we propose the use of Focal Loss [3] as a loss function to deal with the data imbalance. After experimenting and evaluating, our model indicates that it can ascertain malicious software on data streams, beating the MLP, SVN, and Random Forest contest baselines. In the end, our model achieves a TPR (True Positive Rate) 99.56% and 0.49% FAR (False Alarm Rate) on the top-level classification task, and a TPR (True Positive Rate) 76.30% and mAP (mean average precision) 0.4475 on the fine-grained classification task.**

***Index Terms*—Convolutional Neural Network, network traffic analysis, Focal Loss, malware, imbalanced data**

I. Introduction

At present, with the extensive use adoption of devices connected to the Internet and the value of the data on them, one of the most common threats to Internet Security is the emergent spread of malicious software. It is possible to ascertain malicious software searching for patterns on executable code or analyzing its behavior. The NetML Network Traffic Analytics Challenge 2020 aims to facilitate the experimentation and traffic analysis providing a common ground to network traffic analysis with a collection of datasets and a benchmarking site for publishing and compare results [1]. In this paper, we engrossed on the NetML Network Traffic Analytics Challenge 2020 malware detection track. This track is divided into: first, a top-level classification, a binary classification among benign and malicious software generated data stream; second, a fine-grained classification, in which it is required to distinguish among twenty malware classes and a benign class. The contributions of this work can be summarized as follows:

- Show an approach to the malware detection task as an image-processing task, preprocessing the numeric features to dispose them into a 2D image structure.
- Present how a Convolutional Neural Network [2] can be utilized to ascertain patterns on meta-data from network-data streams and discriminate among benign and malware generated streams.
- Show the advantage of Focal Loss [3] use in conjunction with a Convolutional Neural Network for malware detection due to common data imbalance in this type of task.

This paper is comprised of five sections: the first one presents an overview of this task and study. The second section describes some preceding related work to malware detection and related work to the use of images for visualization and classification. The third section describes the problem and used dataset. The fourth section presents some informational background of blocks used to build the proposed system. The fifth section describes the data preprocessing required to feed the Convolutional Neural Network to proper operation as a malware classifier. The sixth section describes the system architecture proposed. The seven section demonstrates the results attained in the experimentation. The last section presents some conclusions and future work to continue the experiments on this task.

II. Related Work

Diverse Machine Learning techniques for malware detection have been proposed, the most common data-driven oriented methods groups similar data samples in families, and let the algorithms learn family-common features to discriminate amongst families [4]. Autoencoders [5], support vector machines (SVMs) [6], and random forest ensembles [7] are some of the proposed methods already for the malware detection task. Related to the utilization of images for the malware identification task have been shown that malware behavior like API calls visualized as color images can be used to identify distinct malware variants [8]. Another preceding method uses

This work was supported by Facultad de Ingeniería de Sistemas, Informática y Ciencias de la Computación (FISICC) and Research Laboratory in Information and Communication Technologies (RLICT), both part of Universidad Galileo from Guatemala.

a Convolutional Neural Network to extract features from malicious software executable code, preprocessing the executable code as a grayscale image for the Convolutional Neural Network input [9]. This method uses a bio-inspired optimization algorithm (BAT) [10] to deal with the data imbalance. Use of Convolutional Neural Networks to evaluate sequences of API calls from Android packets to malware detection on Android has been proposed too [11].

III. Problem Statement

A. Problem Description

The primary objective of this task is to detect malware flows on network traffic through routers. This task is divided into: first, a binary classification task, using top-level annotations containing benign or malware classes; second, a fine-grained task, using fine-grained annotations containing twenty different malware types and a benign class [1].

B. Dataset Description

The dataset is created from raw traffic data captured from a Stratosphere IPS. The raw captured files from Stratosphere IPS are used to extract four sets of features: Meta-data, TLS, DNS, and HTTP. Due to TLS, DNS and HTTP features are not present in all captured samples, we only use meta-data features. Meta-data features are statistical features or histograms protocol-independent as packet numbers, inbound and outbound bytes, and time length, among others [1].

IV. Background

For a better understanding, in this section, we presented some conceptions that explain the focal blocks used to build our model. The following concepts are introduced: Convolutional Neural Networks [2] and Focal Loss [3].

A. Convolutional Neural Networks

Convolutional Neural Networks [2] are Deep Learning models primarily used successfully on the image recognition task [12]. The most imperative characteristic of this type of networks is its capacity to act as automatic feature extractors, with a minimum preprocessing of the input data, removing the needed for hand-engineered features. Furthermore, removes the requirement of a separate feature extractor [13], being capable of being trained jointly with a classifier stage. To attain that, these networks combine three main ideas: local receptive fields, shared weights, and a complete or partial subsampling [13].

B. Focal Loss

The NetML dataset is imbalanced, causing that classes are not equally represented across all the data; this causes a problem in which models are biased to the majority class. The Focal Loss function can deal with the class imbalance problem, modifying the cross-entropy loss, using a scale factor that decays to zero as confidence in the correct class increases. Dynamically, on the training phase, the scale factor down-weight the contribution of easy to classify examples to focus on hard to classify examples [3].

V. Data Preprocessing

We generate an image for each sample. Each sample from the NetML dataset will cause an 11x11-pixel image due to NetML original dataset has 121 metadata features. We took the numerical values and scale them to certify having a 0-255 range of values only for visualization purposes. Later the images are normalized to 0-1 range to feed the model to help the optimization algorithm [14]. We experimented drawing images placing the features in sequential order, but due to Convolutional Neural Networks, use receptive fields to identify patterns in data the order of features have an impact in the classification performance [15]. Therefore, we manually clustered the related features, mainly those that are histograms, to assist the Convolutional Neural Network to discover patterns in a better manner.

VI. Systems Description

Convolutional Neural Networks have been proved a successful architecture to extract imperative features from images on image recognition and classification tasks [12]. It has been shown that the use of visualization speeds up the malware detection process significantly [16]. Based on recent research results, two Convolutional Neural Networks based architectures with slightly altered structures, to process meta-data features in a grayscale 2D representation are proposed. A first model Fig. 1 is presented for the binary malware identification task (malware, benign class) and a second model Fig. 2 is proposed for the multi-class malware identification task. The first model inspired by VGG16 architecture [17] is comprised of a block of three convolutional layers of 256 filters, with a kernel size of 2x2 followed by a max-pooling layer to feed a group of three convolutional layers of 512 filters with a kernel size of 1x1. Two Fully Connected layers of 4096 units follow the entire previous stack, and the last layer is a sigmoid layer to perform binary classification. For the multi-class task, the architecture is comprised of three blocks of two convolutional layers each, with 64, 128, 256 filters respectively, all with a kernel size of 2x2. After the preceding convolutional stack, an average-pooling layer is used following two convolutional layers of 512 filters and a kernel size of 2x2 each with a batch normalization layer in the middle. In the end, a global average pooling layer is used instead of a Fully Connected layer stack to reduce overfitting [18]. This last layer feeds a softmax layer directly for output class classification. For the two architectures, ReLU [12] is used as an activation function on convolutional layers.

The models were trained using an Adam optimizer (learning rate = 3e-5, epsilon = 1e-8) with a batch size = 64. For the first model, the binary focal loss (alpha=.25, gamma=2) was used as a loss function, and for the second model, sparse categorical focal loss (gamma=2) was used. The first model was trained for 51 epochs and the second was trained for 340 epochs.

VII. Results

The official competition metric for binary classification task was TPR (True Positive Rate) and FAR (False Alarm

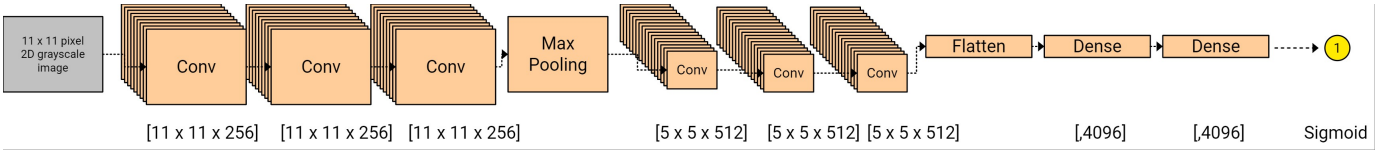

Fig. 1. Convolutional architecture for top-level malware detection.

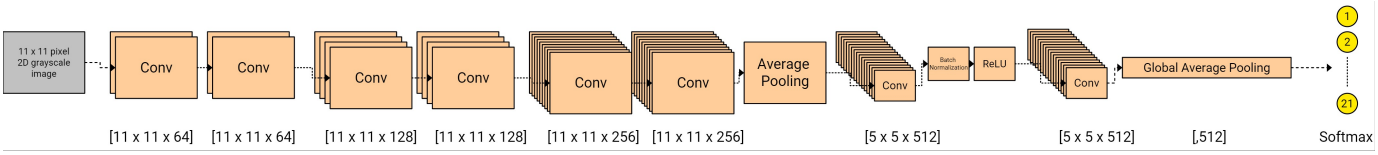

Fig. 2. Convolutional architecture for fine-grained malware detection.

Rate). For the multi-class classification task, the metrics were TPR (True Positive Rate) and mAP (mean average precision) respectively. Table I and Table II indicates our results on NetML Network Analytics Challenge 2020 on the development test sets, for top-level and fine-grained tasks respectively. Additionally, the outcomes of SVG, Random Forest, and MPL baselines provided by organizers are included for comparison.

TABLE I
TOP-LEVEL DEVELOPMENT PHASE RESULTS.

Team Name	TPR	FAR	Overall
UGalileo (CNN + Focal Loss)	**0.99612**	**0.00653**	**0.98961**
Baseline (Random Forest)	0.99365	0.00916	0.98455
Baseline (MLP)	0.98869	0.01706	0.97182
Baseline (SVM)	0.96239	0.01369	0.94921

TABLE II
FINE-GRAINED DEVELOPMENT PHASE RESULTS.

Team Name	F1	mAP	Overall
UGalileo (CNN + Focal Loss)	**0.76130**	**0.43463**	**0.33089**
Baseline (Random Forest)	0.74419	0.42171	0.31384
Baseline (MLP)	0.73138	0.41163	0.30105
Baseline (SVM)	0.69592	0.35362	0.24609

Based on the results, our proposed architectures achieve higher scores compared with the baselines, and we can observe that such architectures can identify malware in binary and multi-class tasks encoding meta-data features as 2D grayscale images.

VIII. Conclusions and Future Work

In this work, we described a Convolutional Neural Network-based architecture to be used in a malware detection task. According to our results, we show that it is possible to ascertain malware classes using meta-data from network streams using image-processing techniques. Our experimental results indicate that the use of Focal Loss [3] can be used on this type of dataset when data is imbalanced commonly. Additionally, it would be worthy of continuing researching possible arrange configurations for the data in the preprocessing phase, when data is transformed from flat-vector structure to a 2D structure, to find an optimal placing for features in such way that Convolutional Neural Network can identify patterns in a better way.

References

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

[2] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, 1999.

[3] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[4] T. Chakraborty, F. Pierazzi, and V. S. Subrahmanian, "Ec2: Ensemble clustering and classification for predicting android malware families," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 262–277, 2020.

[5] A. Naway and Y. Li, "Android malware detection using autoencoder," *ArXiv*, vol. abs/1901.07315, 2019.

[6] T. Singh, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "Support vector machines and malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 12, no. 4, pp. 203–212, 2016. [Online]. Available: https://doi.org/10.1007/s11416-015-0252-0

[7] S. A. Roseline, A. D. Sasisri, S. Geetha, and C. Balasubramanian, "Towards efficient malware detection and classification using multilayered random forest ensemble technique," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–6.

[8] S. Z. Mohd Shaid and M. A. Maarof, "Malware behavior image for malware variant identification," in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014, pp. 238–243.

[9] N. P. Poonguzhali, T. Rajakamalam, S. Uma, and R. Manju, "Identification of malware using cnn and bio-inspired technique," in *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2019, pp. 1–5.

[10] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Studies in Computational Intelligence*, p. 65–74, 2010.

[11] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Maldozer: Automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, pp. S48 – S59, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287618300392

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

[13] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time Series*. Cambridge, MA, USA: MIT Press, 1998, p. 255–258.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on*

Machine Learning - Volume 37, ser. ICML'15. JMLR.org, 2015, p. 448–456.

[15] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, "Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Scientific Reports*, vol. 9, 2019.

[16] D. Lee, I. S. Song, K. J. Kim, and J. Jeong, "A study on malicious codes pattern analysis using visualization," in *2011 International Conference on Information Science and Applications*, 2011, pp. 1–5.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[18] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2014.