Random CapsNet Forest Model for Imbalanced Malware Type Classification Task

Aykut Çayır*, Uğur Ünal, Hasan Dağ
Management Information Systems Department, T.C. Kadir Has University, Istanbul, Turkey

Abstract
Behavior of a malware varies with respect to malware types. Therefore, knowing type of a malware affects strategies of system protection softwares. Many malware type classification models empowered by machine and deep learning achieve superior accuracies to predict malware types. Machine learning based models need to do heavy feature engineering and feature engineering is dominantly effecting performance of models. On the other hand, deep learning based models require less feature engineering than machine learning based models. However, traditional deep learning architectures and components cause very complex and data sensitive models. Capsule network architecture minimizes this complexity and data sensitivity unlike classical convolutional neural network architectures. This paper proposes an ensemble capsule network model based on bootstrap aggregating technique. The proposed method are tested on two malware datasets, whose the-state-of-the-art results are well-known.

Keywords: Capsule Networks, Malware, Ensemble, Deep Learning

1. Introduction

Malware type classification is important as much as malware detection problem because system protection softwares are able to decide their strategies with respect to malware family types. Malware families have different behavior and affects on a computer system. For instance, each

*Corresponding author
Email address: aykut.cayir@khas.edu.tr (Aykut Çayır)
malware family uses different resources, files, ports and other components of operating systems. In addition to this, because of trends in technology, new malware types have occurred.

Malware type classification is the most common problem in cybersecurity domain, because strategies of protection systems vary with respect to malware family type. Malware type classification can be done via three different ways such as static, dynamic and image based [1,2,3]. This paper focuses on image based malware family classification problem. In order to do that, two important datasets have been used and obtained results have been compared other models in the literature.

This paper proposes a new model named random capsule network forest (RCNF) based on bootstrap aggregation (bagging) ensemble and capsule network (CapsNet) [4,5]. The main assumption behind the proposed method is to reduce variance of different CapsNet models (as weak learners) using bagging. In this perspective, the main contributions of this paper can be listed as bellow:

- This paper introduces the first application of CapsNet on malware type classification.
- The first ensemble model of CapsNet is presented in the paper.
- The proposed model uses simple architecture engineering instead of complex convolutional neural network architectures and domain specific feature engineering techniques.

This paper also presents a detailed comparison of proposed model with other models in the literature.

The paper is organized as follows, section 2 presents a literature survey for CapsNet applications and previous malware analysis studies. Datasets used in the paper are described in section 3. Section 4 gives details of inspiring model and the proposed model. Results are investigated in section 5. The paper is concluded by section 6.
2. Related Work

Convolutional neural networks (CNNs) are very popular and successful models for image based datasets. As expected in visual static malware analysis, there are many CNN architectures. CapsNet, as a new CNN structure, has been implemented in 2017\cite{5}. Especially in health domain\cite{6}, CapsNet have many applications. For instance Afshar et al.\cite{7} uses CapsNet for brain tumor classification problem like classification of breast cancer histology images\cite{8}. Mobiny et al\cite{9} creates a fast CapsNet architecture for lung cancer diagnosing. Another important application area of CapsNet is object segmentation. LaLonde et al.\cite{10} uses CapsNet for object segmentation. The most brilliant idea in machine learning is generative adversarial networks (as known as GAN). GANs are based on CNN structures. CapsNet is very useful to make GANs better removing weakest point of CNNs\cite{11}. In other words, these studies show that CapsNet is a promising architecture against to standard CNN.

In the literature, although there are many applications of CapsNets, there is a missing and important area. This area is computer and information security. This gap can be seen easily in a pre-print version of a survey about CapsNets\cite{12}. In that manner, this paper aims to a robust malware prediction model based on CapsNet architecture for imbalanced malware dataset. This model is the first application of CapsNets in malware prediction domain.

3. Datasets

The base CapsNet and the proposed RCNF models have been tested on two the most known malware datasets called Malimg and Microsoft Malware 2015 (BIG2015) respectively. This section describes these two datasets.

3.1. Malimg

Nataraj et al. introduced a new malware family type classification approach based on visual analysis and they had published a new malware dataset\cite{13}. This dataset has 9339 samples and 25 different classes. Table\cite{1} presents the number of samples for each malware family. This distribution shows that the dataset is highly imbalanced.
Figure 1 shows that malware images created from byte files. All images are single channel and have been resized to 224 x 224 for CapsNet architecture. This size is the largest value which can be processed in our computer system.

3.2. Microsoft Malware 2015 (BIG2015)

BIG2015 dataset has been released as a Kaggle competition [14], [15]. Table 2 presents the sample distribution for each malware family in BIG2015 dataset. The distribution shows that the dataset is highly imbalanced and Simda is the toughest malware family to be predicted for the dataset. The dataset contains 10868 BYTE (bytes) files and 10868 ASM (assembly code) files and 9 different malware family types.

Figure 2 depicts image representations creating from BYTE and ASM files of the same malware sample in Ramnit malware family. All images are single channel. All images have been resized 112 x 112 for our CapsNet architecture, because the architecture uses both BYTE and ASM image representations at the same time.
4. Model

In this section, general capsule networks, base CapsNet architecture for Malimg and base CapsNet architecture for Microsoft Malware 2015 Dataset are described. CapsNet architectures are different for both Malimg and BIG2015 Dataset.

4.1. Capsule Networks

Capsule networks are special architectures of convolutional neural networks aiming to minimize information loss because of max pooling \[5\]. These method is the weakest point for preserving spatial information \[8\]. A CapsNet contains capsules similar to autoencoders \[16, 5\]. Each capsule learns how to represent an instance for a given class. Therefore, each capsule creates fixed length feature vector to be input for a classifier layer without using max pooling layers in its internal structure. In this way, this capsule structure aims to preserve texture and spatial information with minimum loss.

Sabor et al. proposes an efficient method to train CapsNet architectures \[5\]. This method is called dynamic routing algorithm. This algorithm uses a new non-linear activation function called squashing shown in equation (1). This equation emphasize that short vectors is shrunk to almost zero and long vectors is shrunk to 1 \[5\]. In this equation, \(v_i\) is the output of ith capsule and \(s_i\) shows the total input of this capsule.
$$v_i = \frac{||s_i||^2}{1 + ||s_i||^2} \frac{s_i}{||s_i||}$$  \hfill (1)

Visualizing of the squash activation function described in equation (1) is very hard because its input is a high dimensional vector. If the activation function can be thought as a single variable function as described in [17] then the behavior of function and its derivative can be visualized in figure [3].

A basic CapsNet architecture contains two parts such as standard convolution blocks and capsule layer as shown in figure [4]. A convolution block is made from combination of convolution filters and ReLU activation function. At the end of the convolution block, obtained feature maps are reshaped and projected to d-dimensional vector representation. This representation feeds each capsule in the capsule layer. Each capsule learns how to represent and reconstruct a given sample like an autoencoder [16] architecture. This representations are used to calculate the class probabilities for classification task.

Margin loss function is used for CapsNet. This function is similar to hinge loss [18]. Equation (2) defines the margin loss $L_c$ for capsule $c$,

$$L_c = y_c \times (\max(0, m - \hat{y}_c))^2 + \lambda \times (1 - y_c) \times (\max(0, \hat{y}_c - (1 - m)))^2 \hfill (2)$$

where $m = 0.9$, $\lambda = 0.5$, $y_c$ denotes actual class and $\hat{y}_c$ represents the current prediction. The summation of $L_c$ for each capsule gives the total loss. In order to minimize margin loss, the most applicable optimizer algorithm for CapsNet is Adam [19,5]. We have observed that CapsNet cannot converge to minimum loss value with other optimizer than Adam. This is obviously an open issue for CapsNet studies in the future.
Our main assumption is that CapsNet architecture will be able to successfully classify malware family types using raw pixel values obtained from malware binary and assembly files. In addition to the main assumption, this paper aims to increase CapsNet malware type classification architecture accuracy with bagging ensembling method.
4.2. Base Capsule Network Model for Malimg Dataset

Before creating an ensemble CapsNet model, base CapsNet estimator must be built. This architecture depends on the dataset. Thus, base CapsNet estimator architecture has a single convolution line as shown in figure 5. The convolutional line contains two sequential block and each block contains two sequential convolution and ReLU layers. First two convolutional layers have 3x3 kernels and 32 filters. Second two convolutional layers have 3x3 kernels and 64 filters. Feature maps are reshaped to 128 feature vector. At the end of the reshape step, there is a capsule layer containing 25 capsules, dimension of each capsule is 8 and the routing iteration is 3 of the capsule layer. This is the optimal CapsNet architecture for Malimg dataset depending on our experiments.

4.3. Base Capsule Network Model for BIG2015 Dataset

The BIG2015 dataset has two different files for each sample. One of them is a binary file and the other is assembly file. Thus, it is possible to
design a CapsNet, which can be fed by two different image inputs at the same time. Figure 5 shows a CapsNet architecture, which has two exactly identical convolution lines. In this architecture, the first two sequential layers contain 3x3 kernels and 64 filters. The second two sequential layers contain 3x3 kernels and 128 filters.

Features extracted from ASM image and BYTE image are concatenated and the final feature vector is reshaped to a vector which length is 128. For the next level, as an input, this feature vector feeds to a capsule layer containing 9 capsules. In this layer, dimension of each capsule is 8 and routing iteration is 3. This hyper-parameter set is optimal for the base CapsNet estimator for BIG2015 dataset.
Random CapsNet Forest (RCNF) is an ensemble model which is inspired from Random Forest algorithm [20]. The basic idea behind RCNF is to assume identical CapsNet models as weak learners and create different training sets for each model from the original training set using bootstrapping resampling technique, as shown in algorithm (1). The training algorithm (1) is a variant of bootstrap aggregating (as known as bagging) [4] for CapsNet model and bagging reduces variance of the model and increases robustness of the model [21]. In this paper, bagging is preferred to create ensemble of CapsNet instead of boosting [22], because it is shown that boosting tends to be overfit [23]. During the training phase, each epoch updates weights of the CapsNet. Therefore, the weight of the best model at the end of each epoch is saved according to validation score in order to increase model performance and consistency against to random weight initialization of the CapsNet.
| No. | Family Name          | Number of Samples |
|-----|----------------------|-------------------|
| 1   | Allaple.L            | 1591              |
| 2   | Allaple.A            | 2949              |
| 3   | Yuner.A              | 800               |
| 4   | Lolyda.AA 1          | 213               |
| 5   | Lolyda.AA 2          | 184               |
| 6   | Lolyda.AA 3          | 123               |
| 7   | C2Lop.P              | 146               |
| 8   | C2Lop.gen!g          | 200               |
| 9   | Instantaccess        | 431               |
| 10  | Swizzot.gen!I        | 132               |
| 11  | Swizzor.gen!E        | 128               |
| 12  | VB.AT                | 408               |
| 13  | Fakerean             | 381               |
| 14  | Alueron.gen!J        | 198               |
| 15  | Malex.gen!J          | 136               |
| 16  | Lolyda.AT            | 159               |
| 17  | Adialer.C            | 125               |
| 18  | Wintrim.BX           | 97                |
| 19  | Dialplatform.B       | 177               |
| 20  | Dontovo.A            | 162               |
| 21  | Obfuscator.AD        | 142               |
| 22  | Agent.FYI            | 116               |
| 23  | Autorun.K            | 106               |
| 24  | Rbot!gen             | 158               |
| 25  | Skintrim.N           | 80                |

Table 1: Sample Distribution for each Malware Family
| No. | Family Name     | Number of Image |
|-----|----------------|-----------------|
| 1   | Ramnit         | 1541            |
| 2   | Lollipop       | 2478            |
| 3   | Kelihos_ver3   | 2942            |
| 4   | Vundo          | 475             |
| 5   | Simda          | 42              |
| 6   | Tracur         | 751             |
| 7   | Kelihos_ver1   | 398             |
| 8   | Obfuscator.ACY | 1228            |
| 9   | Gatak          | 1013            |

Table 2: Number of Sample for each Malware Family in BIG2015 Dataset
**Algorithm 1** Random CapsNet Forest Training Algorithm

1: procedure TRAIN(base_model, n_estimators, trainset, valset, epochs)
2:     for $i \leftarrow 1, n_{\text{estimators}}$ do
3:         $bs_{\text{trainset}} \leftarrow \text{resample}(\text{trainset}, \text{replacement} = \text{True})$
4:     for $e \leftarrow 1, \text{epochs}$ do
5:         $\text{base}_{\text{model}}.\text{fit}(bs_{\text{trainset}})$
6:         $\text{val}_{\text{score}} \leftarrow \text{get\_accuracy}(\text{base}_{\text{model}}, \text{valset})$
7:         if is_best_score(val_score) == True then
8:             save_weights(base_model)

**Algorithm 2** Random CapsNet Forest Prediction Algorithm

1: procedure PREDICT(n_estimators, testset, numclasses) ↦ Average Ensembling
2:     $\text{total\_preds} \leftarrow \text{zeros\_like}(\text{testset}\_\text{shape}[0], \text{numclasses})$
3:     for $i \leftarrow 1, n_{\text{estimators}}$ do
4:         $\text{model}_i \leftarrow \text{load\_model\_weights}(i)$
5:         $\text{total\_preds} += \text{model}_i.\text{predict}(\text{testset})$
6:     $\text{preds} \leftarrow \text{total\_preds} / n_{\text{estimators}}$
7:     return argmax(preds) ↦ The final predictions of CapsNet models
The prediction method is described in algorithm (2). Each weight of CapsNet model is loaded and test samples are predicted by the model. Cumulative predicted probabilities are added into total_preds variable and this step is known as average ensembling step. At the end of the estimation loop, the index of the highest probabilities is assigned as predicted class.

5. Experiment and Results

CapsNet and RCNF ensemble model are tested on two different datasets called Malimg and BIG2015. Malimg dataset has been divided into three parts, training, validation and test sets. Training set has 7004 samples, validation set has 1167 samples and test set has 1167 samples. BIG2015 has also been divided into three parts like Malimg dataset. In the experiments of CapsNet and RCNF ensemble model for BIG2015 dataset, training set has 8151 samples, validation set has 1359 samples and test set has 1358 samples.

The First experiment is to obtain performance results of single base CapsNet estimators for each dataset. The second experiment is about performance of RCNF model. Model evaluation has been done in terms of accuracy, $F_1$ measure and number of parameters of deep neural nets. These performance metrics are defined as follows:

\[
\text{accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3}
\]

\[
F_1 = \frac{2 \times TP}{2 \times TP + FN + FP} \tag{4}
\]

where true positive (TP) and false positive (FP) are the number of instances correct and wrong classified as positive, respectively. True negative (TN) and false negative (FN) are the number of instances correct and wrong classified as negative, respectively. Accuracy is the ratio of number of true predictions to all instances in the set as shown in equation (3). $F_1$ is shown as the equation (4) in terms of true positives, false negatives and false positives.

Figure 7 shows confusion matrices for each test part of both datasets. Each confusion matrix (figure 7a and 7b) implies that a model containing single CapsNet incorrectly predicts rare malware families in both datasets.
Figure 8 is the confusion matrix of RCNF containing 5 base CapsNet models. This confusion matrix shows prediction accuracy of the model for each malware family type in Malimg test set. Class 8, 10, 20 and 21 have been predicted wrongly by the RCNF model. On the other hand, the model has been very successful at correctly predicting other malware types in the test set. This confusion matrix also shows that RCNF is successful at correctly predicting rare malware types in Malimg test set.

Figure 7: Confusion Matrices of Single CapsNet Model for each Test Sets

(a) Malimg Test Set
(b) BIG2015 Test Set
In the second experiment, RCNF is tested on BIG2015 dataset. Figure 8 shows the prediction results of RCNF containing 10 base CapsNet for BIG2015 dataset. Class 4 is the rarest malware type in the whole dataset. Training, validation and test sets are stratified, so the class distribution is preserved for each partition. This result shows that RCNF is able to predict the rarest malware type pretty well. Class 0, 1, 2 and 6 are predicted perfectly by RCNF. If the performance of RCNF is compared with the performance of a single CapsNet model, it is easily seen that RCNF is better than a single CapsNet at predicting rare malware families for imbalanced datasets.
Table 3: Comparison RCNF and other Methods for Malimg Test Set Performance

| Model              | Number of Parameters | F1 Score | Accuracy |
|--------------------|----------------------|----------|----------|
| Yue [24]           | 20M                  | -        | 0.9863   |
| Cui et al. [25]    | -                    | 0.9455   | 0.9450   |
| CapsNet for Malimg | 90592                | 0.9658   | 0.9863   |
| RCNF for Malimg    | 5 x 90592            | 0.9661   | 0.9872   |

Table 3 shows the test performance of the proposed models and others for Malimg dataset. Yue [24] uses a weighted loss function to handle im-
Table 4: Comparison RCNF and other Methods for BIG2015 Test Set Performance

| Model                  | Number of Parameters | F1 Score | Accuracy |
|------------------------|----------------------|----------|----------|
| Cao et al. [26]        | -                    | -        | 0.95     |
| Gibert et al. [27]     | -                    | 0.9813   | 0.9894   |
| Kreuk et al. [28]      | -                    | -        | 0.9921   |
| Le et al. [29]         | 268949               | 0.9605   | 0.9820   |
| Chen [30]              | -                    | -        | 0.9925   |
| Jung et al. [31]       | 148489               | -        | 0.99     |
| Abijah et al. [2]      | -                    | 0.9914   |          |
| Zhao et al. [32]       | -                    | -        | 0.929    |
| Khan et al. [33]       | -                    | -        | 0.8836   |
| Safa et al. [?]        | -                    | -        | 0.9931   |
| Kebede et al. [34]     | -                    | -        | 0.9915   |
| Kim et al. [35]        | -                    | -        | 0.9266   |
| Kim et al. [36]        | -                    | 0.8936   | 0.9697   |
| Yan et la. [37]        | -                    | -        | 0.9936   |
| Naeem et la. [38]      | -                    | 0.971    | 0.9840   |
| CapsNet for BIG2015    | 527232               | 0.9779   | 0.9926   |
| RCNF for BIG2015       | 10 x 527232          | 0.9820   | 0.9956   |

balance class distribution problem in Malimg dataset and also uses transfer learning [39] method to classify malware family types. Due to using transfer learning, the architecture has 20M parameters and the model is so huge. Cui et al. [25] use classical machine learning methods such as K-Nearest Neighbor and support vector machines. They have trained these algorithms using GIST and GLCM features which are feature engineering methods for images and have applied resampling to the dataset in order to solve imbalance dataset problem. RCNF does not use weighted loss function or any sampling method to overcome imbalanced dataset problem. Our results are higher than these two methods and results also show that CapsNet and RCNF do not require any method for extra feature engineering in Malimg dataset. A single CapsNet architecture for Malimg
dataset has 90592 trainable parameters and RCNF has 452960 trainable parameters, so our proposed methods are reasonably smaller than Yue’s method.

Table 4 compares the test performance of the proposed models and others for BIG2015 dataset. Chen [30] and Khan et al. [33] use transfer learning architectures for the dataset. Test results of our proposed methods are better than those two models, but our results are very close to Chen [30] in terms of accuracy. Our proposed models are better than Gibert et al. [27] in terms of accuracy, but F1 score of RCNF is very close to this model. Our proposed models are obviously better than models of Cao et al. [26], Zhao et al. [32], Kim et al. [35] and Kim et al. [36]. Jung et al. [31] propose a reasonably smaller model than our models in terms of number of parameters, but our model has higher accuracy score than this model. Abijah et al. [2], Safa et al. [40], Kebede et al. [34] and Yan et al. [37] propose deep learning models whose accuracy scores are close to our proposed models. Data resampling and weighted loss function for imbalanced dataset, transfer learning and data augmentation have never been used to train CapsNet and RCNF models for both BIG2015 and Malimg datasets. In other words, these results are basic performances of these models.

For those tables, last studies using Malimg and BIG2015 datasets are chosen. In order to compare in a fair way, these models are drawn from visual static malware analysis studies.
6. Conclusion

This paper introduces a first application of CapsNet on imbalanced malware family type classification task. Moreover, the first ensemble model of CapsNet called RCNF is introduced in this paper. Proposed models do not require any complex feature engineering methods or architecture for deep networks. In order to show that, we used two different malware family type datasets such as Malimg and BIG2015. These datasets are used image based visual static malware classification. Our proposed models can use these datasets directly using raw pixel values.

Datasets in the paper are highly imbalanced in terms of class distribution. CapsNet and RCNF do not use oversampling, undersampling and weighted loss function during training phase. Results show that CapsNet and RCNF are the best models least suffering from imbalanced class distribution among others.

Experiment results show that a single CapsNet model has good performance for both BIG2015 and Malimg datasets. However, we have assumed an ensemble model of CapsNet can help to increase generalization performance and RCNF has better generalization performance results than a single CapsNet model as expected.

As a future work, we are planning to develop a hybrid architecture for malware classification. This hybrid method will be based on CapsNet architecture. This paper shows that CapsNet are useful for imbalanced malware dataset in visual static malware analysis problem domain.
References

[1] L. Nataraj, V. Yegneswaran, P. Porras, J. Zhang, A comparative assessment of malware classification using binary texture analysis and dynamic analysis, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, ACM, pp. 21–30.

[2] S. Abijah Roseline, et al., Intelligent malware detection using deep dilated residual networks for cyber security, in: Countering Cyber Attacks and Preserving the Integrity and Availability of Critical Systems, IGI Global, 2019, pp. 211–229.

[3] S. Ni, Q. Qian, R. Zhang, Malware identification using visualization images and deep learning, Computers & Security 77 (2018) 871–885.

[4] L. Breiman, Bagging predictors, Machine learning 24 (1996) 123–140.

[5] S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, in: Advances in neural information processing systems, pp. 3856–3866.

[6] A. Jiménez-Sánchez, S. Albarqouni, D. Mateus, Capsule networks against medical imaging data challenges, in: Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis, Springer, 2018, pp. 150–160.

[7] P. Afshar, A. Mohammadi, K. N. Plataniotis, Brain tumor type classification via capsule networks, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, pp. 3129–3133.

[8] T. Iesmantas, R. Alzbutas, Convolutional capsule network for classification of breast cancer histology images, in: International Conference Image Analysis and Recognition, Springer, pp. 853–860.

[9] A. Mobiny, H. Van Nguyen, Fast capsnet for lung cancer screening, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, pp. 741–749.

[10] R. LaLonde, U. Bagci, Capsules for object segmentation, arXiv preprint arXiv:1804.04241 (2018).
[11] A. Jaiswal, W. AbdAlmageed, Y. Wu, P. Natarajan, Capsulegan: Generative adversarial capsule network, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 0–0.

[12] M. K. Patrick, A. F. Adekoya, A. A. Mighty, B. Y. Edward, Capsule networks—a survey, Journal of King Saud University-Computer and Information Sciences (2019).

[13] L. Nataraj, S. Karthikeyan, G. Jacob, B. Manjunath, Malware images: visualization and automatic classification, in: Proceedings of the 8th international symposium on visualization for cyber security, ACM, p. 4.

[14] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, M. Ahmadi, Microsoft malware classification challenge, arXiv preprint arXiv:1802.10135 (2018).

[15] Microsoft, Microsoft malware classification challenge (big 2015) — kaggle, https://www.kaggle.com/c/malware-classification 2015. (Accessed on 05/06/2019).

[16] A. Krizhevsky, G. E. Hinton, Using very deep autoencoders for content-based image retrieval., in: ESANN, volume 1, p. 2.

[17] A. Marchisio, M. A. Hanif, M. Shafique, Capsacc: An efficient hardware accelerator for capsules with data reuse, in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, pp. 964–967.

[18] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, A. Verri, Are loss functions all the same?, Neural Computation 16 (2004) 1063–1076.

[19] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[20] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[21] L. Breiman, Bias, variance, and arcing classifiers (1996).

[22] Y. Freund, R. E. Schapire, et al., Experiments with a new boosting algorithm, in: icml, volume 96, Citeseer, pp. 148–156.
[23] J. R. Quinlan, et al., Bagging, boosting, and c4. 5, in: AAAI/IAAI, Vol. 1, pp. 725–730.

[24] S. Yue, Imbalanced malware images classification: a cnn based approach, arXiv preprint arXiv:1708.08042 (2017).

[25] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-g. Wang, J. Chen, Detection of malicious code variants based on deep learning, IEEE Transactions on Industrial Informatics 14 (2018) 3187–3196.

[26] D. Cao, X. Zhang, Z. Ning, J. Zhao, F. Xue, Y. Yang, An efficient malicious code detection system based on convolutional neural networks, in: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, ACM, pp. 86–89.

[27] D. Gibert, C. Mateu, J. Planes, An end-to-end deep learning architecture for classification of malwares binary content, in: International Conference on Artificial Neural Networks, Springer, pp. 383–391.

[28] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, J. Keshet, Deceiving end-to-end deep learning malware detectors using adversarial examples, arXiv preprint arXiv:1802.04528 (2018).

[29] Q. Le, O. Boydell, B. Mac Namee, M. Scanlon, Deep learning at the shallow end: Malware classification for non-domain experts, Digital Investigation 26 (2018) S118–S126.

[30] L. Chen, Deep transfer learning for static malware classification, arXiv preprint arXiv:1812.07606 (2018).

[31] B. Jung, T. Kim, E. G. Im, Malware classification using byte sequence information, in: Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, ACM, pp. 143–148.

[32] Y. Zhao, C. Xu, B. Bo, Y. Feng, Maldeep: A deep learning classification framework against malware variants based on texture visualization, Security and Communication Networks 2019 (2019).

[33] R. U. Khan, X. Zhang, R. Kumar, Analysis of resnet and googlenet models for malware detection, Journal of Computer Virology and Hacking Techniques (2018) 1–9.
[34] T. M. Kebede, O. Djaneye-Boundjou, B. N. Narayanan, A. Ralescu, D. Kapp, Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (big 2015) dataset, in: 2017 IEEE National Aerospace and Electronics Conference (NAECON), IEEE, pp. 70–75.

[35] C. H. Kim, E. K. Kabanga, S.-J. Kang, Classifying malware using convolutional gated neural network, in: 2018 20th International Conference on Advanced Communication Technology (ICACT), IEEE, pp. 40–44.

[36] J.-Y. Kim, S.-B. Cho, Detecting intrusive malware with a hybrid generative deep learning model, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, pp. 499–507.

[37] J. Yan, Y. Qi, Q. Rao, Detecting malware with an ensemble method based on deep neural network, Security and Communication Networks 2018 (2018).

[38] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas, M. S. Javed, Identification of malicious code variants based on image visualization, Computers & Electrical Engineering 76 (2019) 225–237.

[39] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in neural information processing systems, pp. 3320–3328.

[40] H. Safa, M. Nassar, W. A. R. Al Orabi, Benchmarking convolutional and recurrent neural networks for malware classification, in: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, pp. 561–566.