Evaluating Deep Learning Classification Reliability in Android Malware Family Detection

Giacomo Iadarola*, Fabio Martinelli*, Francesco Mercaldo†*, Antonella Santone†
*Institute for Informatics and Telematics, National Research Council of Italy (CNR), Pisa, Italy
{giacomo.iadarola, fabio.martinelli, francesco.mercaldo}@iit.cnr.it
†University of Molise, Campobasso, Italy
{francesco.mercaldo, antonella.santone}@unimol.it

***Abstract*—Artificial intelligence techniques are nowadays widespread to perform a great number of classification tasks. One of the biggest controversies regarding the adoption of these techniques is related to their use as a "black box" i.e., the security analyst must trust the prediction without the possibility to understand the reason why the classifier made a certain choice. In this paper we propose a malicious family detector based on deep learning, providing a mechanism aimed to assess the prediction reliability. The proposed method obtains an accuracy of 0.98 in Android family identification. Moreover, we show how the proposed method can assist the security analyst to interpret the output classification and verify the prediction reliability by exploiting activation maps.**

***Index Terms*—malware, deep learning, reliability, interpretability, explainability, Android, security, artificial intelligence.**

I. Introduction and Motivation

In the last years, we are assisting in increasing adoption of decision support systems based on models exploiting artificial intelligence techniques, typically considering machine or deep learning. These systems are usually considered as black-box [1] aimed to output a prediction i.e., to generate a label for unseen instances. These predicted information are also sensitive and private for instance, models aimed to predict the presence of a cancerous area in magnetic resonance images [2].

The main issue in this context is that artificial intelligence fundamentally lacks a framework for reasoning about failures and their potential effects [3]. This is in contrast to traditional engineering disciplines which have been forced to consider the safety implications across a broad set of applications [4], from building a bridge to the development of secure software systems. As a matter of fact, driven by the success of artificial intelligence in commercial applications (for instance, product recommendations and advertising), there are significant efforts to exploit artificial intelligence tools for data analysing and knowledge extraction [5]. Applying artificial intelligence techniques without careful consideration of their assumptions and limitations may seriously lead to missed opportunities at best and a waste of substantial resources and incorrect scientific inferences at worst [6]. The need for interpretable models, able to provide the meaning of its predictions in understandable terms to humans, is fostering an active research field [7]. Many works propose to explore the model network activations, in order to give "visual explanations" to the model decision. For instance, the paper by Zhou B. et al [8] produces a Class Activation Map (CAM) using global average pooling in the CNN model. The CAM for a particular prediction shows the discriminative area in the input image that identifies that output label. The grad-CAM [9] adopted by this work is a relaxed generalization of the previous approach. A model prediction easy to understand by a human is also a model easy to use by a non-expert. For instance, the work by Le Q. et al [10] presents a model to perform malware classification tasks that require no expert domain knowledge, based on a data-driven approach to identify common patterns to perform the classification. The paper by Ribeiro M.T. et al [11] introduces LIME, a technique to explain the prediction of classifiers, which allow a non-expert to judge models, and then deciding if their predictions are trustable.

One of the most widespread applications of artificial intelligence techniques is concerning malware detection [12] with particular regard to the Android platform [13]. In this context, machine and deep learning are exploited for building models for detecting the maliciousness of Android samples by invoking a model previously built with a set of known applications [14], [15], [16].

Starting from these considerations, in this paper we propose a deep learning network, designed by the authors, for detecting Android malicious families, and also exploiting the activation maps to interpret the model output. We represent Android applications as images, used to input the deep learning network for model building and testing. In the evaluation, once the model predicts the label (i.e., the malicious family) for an unseen Android application, we generate the activation map related to the image of the submitted sample. The activation map shows the areas of the image analysed by the model to predict a certain label; basically, the activation map provides to the security analyst a heatmap to understand where the model is focusing to make a certain prediction. Moreover, once a sample is misclassified it is possible to visually understand how much the activation maps are distant to the ones of the images rightly predicted. Especially in the context of malware represented as images, where the input image may look like random noise for the security analyst, the activation map is relevant to understand the reason why the model makes a certain prediction, and then evaluate the model reliability. For instance, it is expected that malicious samples belonging to

978-1-7281-7735-9/20/$31.00 ©2020 IEEE
DOI 10.1109/ISSREW51248.2020.00082

the same family, in addition to being predicted as belonging to the correct family, exhibit similar activation maps, thus the security analyst can conclude that certain areas of the images are symptomatic of a certain malicious family. This is coherent with the definition of malware family: as a matter of fact, samples belonging to the same family exhibit the same malicious payload [14] (i.e., the same malicious behaviour), that can be related to the areas interested by the activation maps as symptomatic of the malicious payload.

The paper proceeds as follows: in the next section we describe the proposed method for reliable malicious family detection in Android environment, in Section III we evaluate the proposed method exploiting a dataset of real-world Android malware and, finally, in the last section conclusion and future research plans are drawn.

II. The Method

Coherently with supervised machine learning methods, the proposed method consists in two phases: the training phase, depicted in Figure 1, and the testing phase, shown in Figure 3.

The proposed method starts from a set of Android applications with the related labels (i.e., the malware family). Then, each application is represented in terms of an image obtained from the application binary, as shown in Figure 1.

From the Android application we extract the *dex* executable file. To convert the *dex* file to an image we treat the sequence of bytes representing the *dex* binary file as the bytes of a grey-scale PNG image [17], [18]. We consider a predefined width of *256*, and a variable length, depending on the size of the binary. We developed a script to encode a *dex* file into a lossless PNG [19].

The grey-scale image generation considers each byte of the binary file and converts them into numbers (*0-255*), which will then define a pixel colour, and thus a grey-scale pixel in the final PNG image. The script is freely available for research purposes[1].

The grey-scale images, representing the Android malicious applications with the relative family label, are the input for the deep network, which architecture is depicted in Figure 2 and details about the layers are provided in Table I.

Basically the proposed network is composed by three set of 2D convolutional (*Conv2D* in Table I) and MaxPooling layers (*MaxPooling2D* as shown in Table I), followed by several *Flatten*, *Dropout* and *Dense* layers.

The aim of the deep network is to build and train a model able to distinguish the application in input between the several malware families, as shown in the last step of Figure 1.

In Figure 3 we show the evaluation of the model we built in the previous step.

We consider an unseen Android application, obtaining from the application binary the relative grey-scale image and thus we input the model with this image (in this case, clearly, we do not consider the family label): the model will output the family label with a confidence percentage, based on the output percentages of the output vector, and the activation maps i.e., the area in the application grey-scale image that, from the deep network point of view, are symptomatic of the prediction. To draw the activation map we resort to the Gradient-weighted Class Activation Mapping (grad-CAM) [9] algorithm, considering class-specific gradient information flowing into the final convolutional layer of a CNN to produce a localization map of the important regions in the image.

TABLE I: The deep network layers.

Layer	Type	Output shape
01	*InputLayer*	[(None, 300, 300, 1)]
02	*Conv2D*	(None, 298, 298, 64)
03	*MaxPooling2D*	(None, 149, 149, 64)
04	*Conv2D*	(None, 147, 147, 96)
05	*MaxPooling2D*	(None, 73, 73, 96)
06	*Conv2D*	(None, 71, 71, 128)
07	*MaxPooling2D*	(None, 35, 35, 128)
08	*Flatten*	(None, 156800)
09	*Dropout*	(None, 156800)
10	*Dense*	(None, 1024)
11	*Dropout*	(None, 1024)
12	*Dense*	(None, 512)
13	*Dense*	(None, 6)

This mechanism provides the security analyst with a way to understand the reason why the model outputs a prediction. Malware that belong to the same family share parts of the code, hence areas of the images will be similar. By taking this intuition into account, we can evaluate the predictions by looking at the activation maps, which should highlight similar areas for malware belonging to the same family, because those areas should represent the shared code among that family. In short, in the malware detection context, the activation map helps to understand the area of the image symptomatic of a malicious behaviour exhibited by a family. This information is especially helpful in this context, where the malware represented as an image is hardly readable for human eyes.

III. Experimental Evaluation

In this section we show the results obtained from the experimental evaluation of the proposed method.

As stated into the introduction, we experiment the proposed deep learning network with a real-world dataset composed by six Android malware families: Airpush (1170 samples), Dowgin (762), FakeInstaller (2130), Fusob (1274), Jisuit (550) and Mecor (1500).

A total of 7386 real-world Android malware were considered, belonging to six different families exhibiting different malicious behaviours, from the *Jisut* screen-locking ransomware to the *Fusob* ability to steal user private and sensitive information by capturing keyboard input.

The malicious samples were obtained from the Android Malware Dataset repository, a well-known collection of malicious samples with the label family[2].

[1] https://github.com/leeroybrun/Bin2PNG

[2] http://amd.arguslab.org/

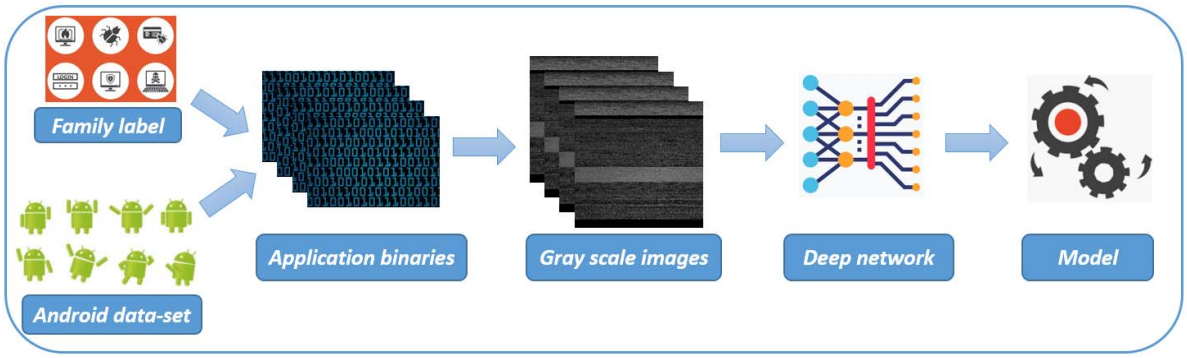

Fig. 1: The training phase.

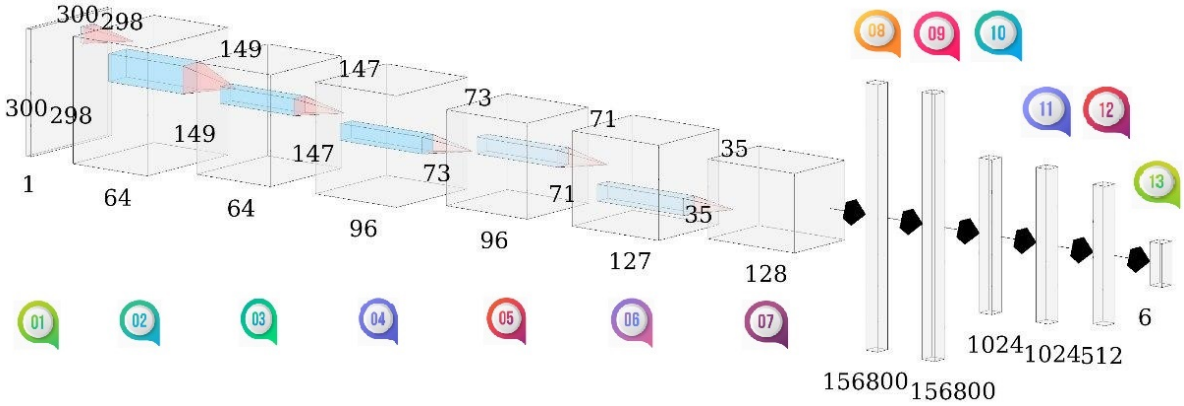

Fig. 2: The deep learning network.

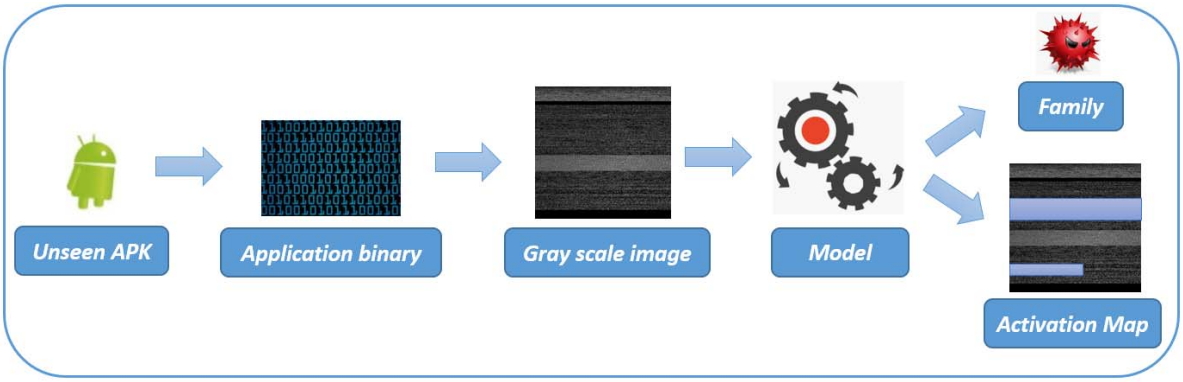

Fig. 3: The testing phase.

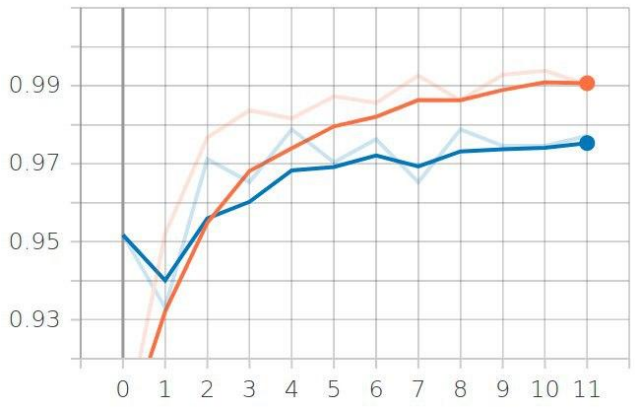

Fig. 4: The accuracy trend, orange for training and blue for validation.

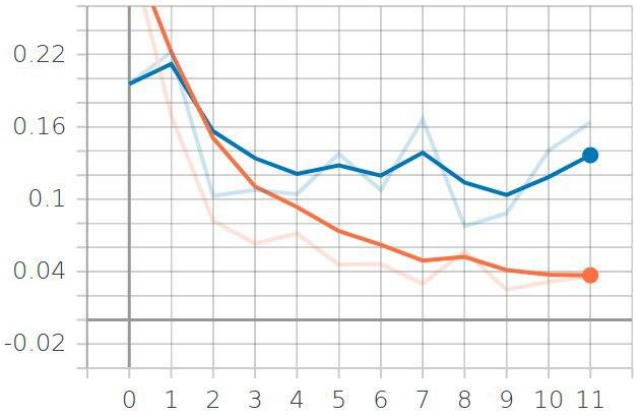

Fig. 5: The loss trend, orange for training and blue for validation.

A. Prediction Performance

We performed a model assessment step to tune the hyper-parameters, then we split the dataset into training and test part in 80:20 ratio for the model building and evaluation steps. The training part of the dataset was split again in 80:20 ratio as training and validation set. We computed the accuracy and loss metrics to evaluate the performances of the classification. In this section are reported only the results produced by the model with the best hyper-parameter selected.

We considered *12* epochs for model building. Figure 4 shows the accuracy for the different epochs, orange line for training and blue line for validation: the trend is growing, suggesting that the network is learning.

Figure 5 shows the loss values for each epoch.

Subsequently the model assessment part and hyper-parameter tuning step, we trained the model over the entire training set again (including the validation set), and then we examined the performances on the test set. The model achieved an accuracy of *0.9843* in test, while the loss value was equal to *0.0767*. Figure 6 shows the confusion matrix, where the

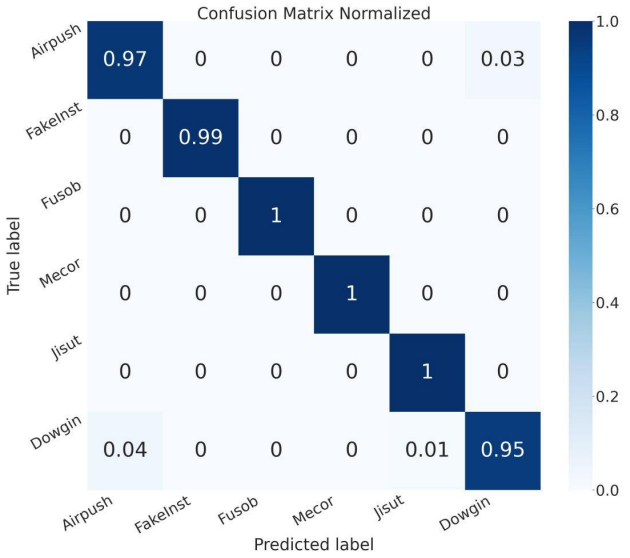

Fig. 6: Confusion matrix.

precision metric is considered to understand the results at a family grain.

The model exhibits a precision equal to 1 for the *Fusob*, the *Mecor* and the *Jisuit* families. A precision ranging from 0.95 to 0.99 is reached for the remaining families: *Dogwin*, *Airpush* and *FakeInstaller*. The grad-CAM can be applied to the trained model to study the outputs as explained in the next subsection

B. Classification Reliability

In this subsection, we explain how the proposed method can be used to understand the classification reliability by discussing some activation map examples. For each sample, we respectively show the grey-scale image of the malware (the model input), the activation map and the picture generated from the overlap of the grey-scale image and the activation map.

Figure 7 shows the activation map for a sample belonging to the *FakeInstaller* family, detected correctly as belonging to the *FakeInstaller* family.

The activation map exhibits an extended dark blue area (not of interest for the model) and several green and yellow areas (symptomatic of the activation).

Figure 8 shows the activation map for another sample belonging to the *FakeInstaller* family, also this one was detected correctly as belonging to the *FakeInstaller* family.

By looking at the two grey-scale pictures of the *FakeInstaller* samples (left part of Figure 7 and 8), we can realize that there is a very characteristic grey area in the medium-low part of the images. At this point in the analysis, the security analyst can make use of the activation maps to visualize which parts of the images were the most important for the classification. The activation maps confirm that the model caught this pattern

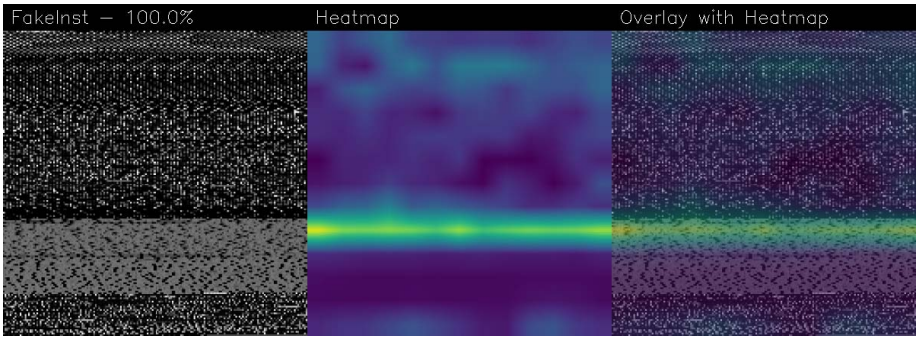

Fig. 7: Sample identified by *7ab97a3d710e1e089e24c0c27496ee76* hash, belonging to the *FakeInstaller* family.

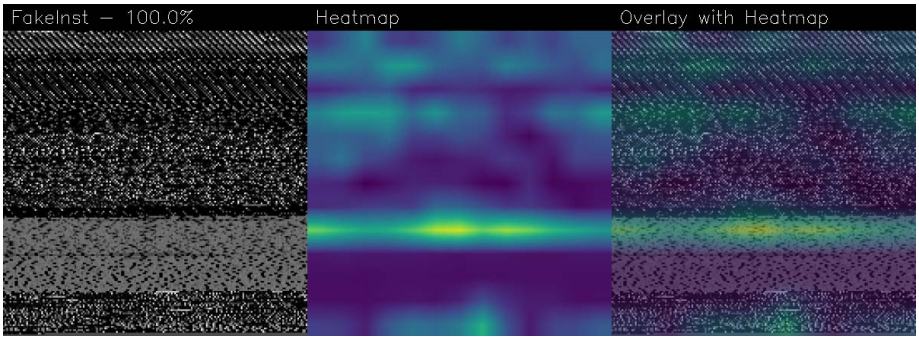

Fig. 8: Sample identified by *0058f6e5225c519df0c0402313061253* hash, belonging to the *FakeInstaller* family.

too, because they highlight exactly those parts of the images. Moreover, both the activation maps in Figure 8 and Figure 7 are similar, thus the security analyst could link that specific pattern to the *FakeInstaller* family. This insight helps the security analyst to interpret the model inference, and extract information from the grey-scale image.

Figure 9 shows the activation map of a sample belonging to the *Airpush* family. The grey-scale image may look similar to the samples of the *FakeInstaller* family (Figure 7 and Figure 8) because of the grey area in the middle-low part of the image. Nevertheless, it belongs to a different family, and the model was able to catch this difference because it classifies the malware rightly; intuitively, they are two different families, hence different areas were peculiar for each family. The grey-scale images are really hard to read for the security analyst and may look like random noise or, even worse, could lead to mistaken similar patterns that do not have any correlation. The activation map plays an important role in helping the security analyst to interpret the classification. Figure 9 shows that the model focused the attention on a different area of the malware with regard of the activation maps of the *FakeInstaller* samples; instead of focusing on the grey middle-low area, the activation map highlights the top area of the image, which seems to be the discriminating and relevant area for the *Airpush* family.

The activation maps are useful also when the model wrongly classifies a sample. Figure 10 shows an example of an activation map belonging to a misclassification. The sample is wrongly detected as belonging to the *Airpush* family, with a confidence of 100%, while it is belonging to the *Dowgin* family.

By comparing the activation maps in Figures 9 and 10, we can observe that they are really different. Even if both samples were classified as belonging to the *Airpush* family, the model focused on different areas for performing the classification. The security analyst can deduce that the *Airpush* family prediction of the sample in Figure 10 is not reliable and further investigations are required.

IV. Conclusion and Future Work

Predictive models considering machine and deep learning are currently really widespread. These models are considered as a black-box aimed to output a prediction for a never seen input instance. Nevertheless, the number of tasks and routines assigned to these models is growing, and so the number of threats, risks and potential damage related. There is a need to interpret predictions and quantify reliability to build robust predictive models. In this paper we propose a method aimed to distinguish between malware families by exploiting deep learning. Moreover, by considering activation maps, we suggest a technique useful for the security analyst to understand whether the prediction can be considered reliable. We evaluate a dataset of 7386 real-world Android malware belonging to six different families, obtaining an average accuracy of 0.98.

Future research plans will take into account an automatic prediction reliability indicator, starting from the analysis of

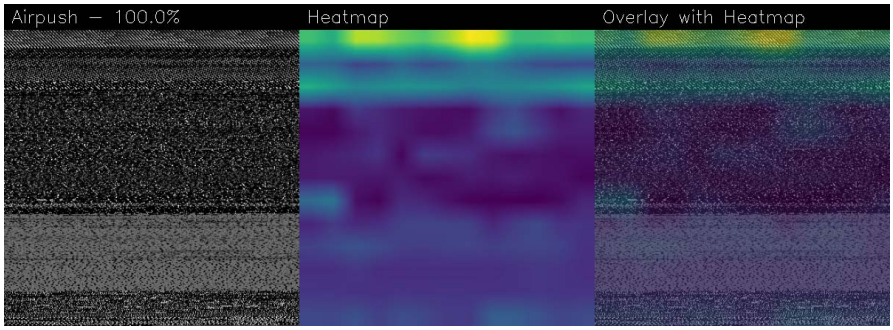

Fig. 9: Sample identified by *7728411fbe86010a57fca9149fe8a69b* hash, belonging to the *Airpush* family.

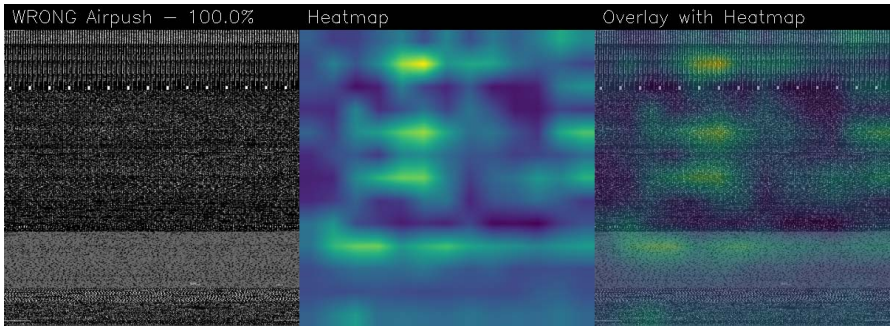

Fig. 10: Sample identified by *9380f4e66af76a10f10a38fd7cb82743* hash, belonging to the *Dowgin* family.

the proposed activation maps. Furthermore, the authors would like to reverse the methodology and investigate the source code highlighted from the activation maps.

ACKNOWLEDGMENTS

This work has been partially supported by MIUR - SecureOpenNets, EU SPARTA contract 830892, CyberSANE project, and the EU project CyberSure 734815.

REFERENCES

[1] D. L. Parnas, "The real risks of artificial intelligence," *Communications of the ACM*, vol. 60, no. 10, pp. 27–31, 2017.

[2] L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "Neural networks for lung cancer detection through radiomic features," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–10.

[3] S. Saria and A. Subbaswamy, "Tutorial: safe and reliable machine learning," *arXiv preprint arXiv:1904.07204*, 2019.

[4] H. Pham, *Software reliability*. Springer Science & Business Media, 2000.

[5] B. Kailkhura, B. Gallagher, S. Kim, A. Hiszpanski, and T. Y.-J. Han, "Reliable and explainable machine-learning methods for accelerated material discovery," *npj Computational Materials*, vol. 5, no. 1, pp. 1–9, 2019.

[6] A. Turchin and D. Denkenberger, "Classification of global catastrophic risks connected with artificial intelligence," *AI & SOCIETY*, vol. 35, no. 1, pp. 147–163, 2020.

[7] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[8] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[9] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[10] Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: Malware classification for non-domain experts," *Digital Investigation*, vol. 26, pp. S118–S126, 2018.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[12] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[13] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, and S. Chen, "A survey on mobile malware detection techniques," *IEICE Transactions on Information and Systems*, vol. 103, no. 2, pp. 204–211, 2020.

[14] G. Canfora, A. De Lorenzo, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Effectiveness of opcode ngrams for detection of multi family android malware," in *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. IEEE, 2015, pp. 333–340.

[15] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Image-based malware family detection: An assessment between feature extraction and classification techniques." in *IoTBDS*, 2020, pp. 499–506.

[16] ——, "Formal methods for android banking malware analysis and detection," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 2019, pp. 331–336.

[17] N. Bhodia, P. Prajapati, F. Di Troia, and M. Stamp, "Transfer learning for image-based malware classification," *arXiv preprint arXiv:1903.11551*, 2019.

[18] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*. ACM, 2011, p. 4.

[19] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, pp. 1–15, 2020.