KOL-4-GEN: Stacked Kolmogorov-Arnold and Generative Adversarial Networks for Malware Binary Classification through Visual Analysis

Anurag Dutta, *Graduate Student Member, IEEE*, Satya Prakash Nayak, Ruchira Naskar, *Member, IEEE*, and Rajat Subhra Chakraborty, *Senior Member, IEEE*

***Abstract*—Malware identification and classification is an active field of research. A popular approach is to classify malware binaries using visual analysis, by converting malware binaries into images, which reveal different class-specific patterns. To develop a highly accurate multi-class malware classifier, in this paper, we propose KOL-4-GEN, a set of four novel deep learning models based on the Kolmogorov-Arnold Network (KAN) with trainable activation functions, and a Generative Adversarial Network (GAN) to address data imbalance (if applicable) during training. Our models, tested on the standard `Malimg` (grayscale, imbalanced, 25 classes), `Malevis` (RGB, balanced, 26 classes), and the miniature `Virus-MNIST` (grayscale, imbalanced, 10 classes) datasets, outperform state-of-the-art (S-O-T-A) models, achieving $\approx 99.36\%$, $\approx 95.44\%$, and $\approx 92.12\%$ validation accuracy, respectively.**

***Index Terms*—Malware classification, Digital Image Forensics, Kolmogorov-Arnold Network (KAN), Generative Adversarial Network (GAN).**

I. INTRODUCTION

MALWARE is a piece of binary code (or a program) that if executed, can potentially harm a system by surpassing its security mechanisms, leading to severe losses for individuals, organizations, and sometimes government bodies. While defence mechanisms like antivirus, firewalls, etc. exist to detect and prevent adversaries from implanting malware in any system, the surge in new malware variants and overriding features of existing malware, makes malware forensics an active research domain. Among the key components of an effective malware defence, are the detection of whether an incoming program is *malicious* or *benign* and if malicious, identification of the malware's class. Given the surreptitious and widely varying *modus operandi* of malware, classifying malware into different classes is quite a challenging task. For example, *Worms* typically replicate and contaminate across networks, but they can also be modified by adversaries to collect sensitive user data and personal information, thus behaving like *Spyware*.

This work primarily focuses on classifying the malware into their classes through visual analysis, i.e. by automated image processing and computer vision techniques. **The main insight behind such approaches is that the image representations of malware (technically, binaries of the malware executable) of the same class exhibit some spatio-temporal similarity and patterns, since malware binaries commonly make repeated use of certain instructions [1].** By considering bytes in the program binaries to represent pixels, and careful analyses of the resulting images, these patterns can be identified for classifying malware. Fig. 1 shows the grayscale image representation of few select malware variants (out of twenty-five total variants) from the `Malimg` database.

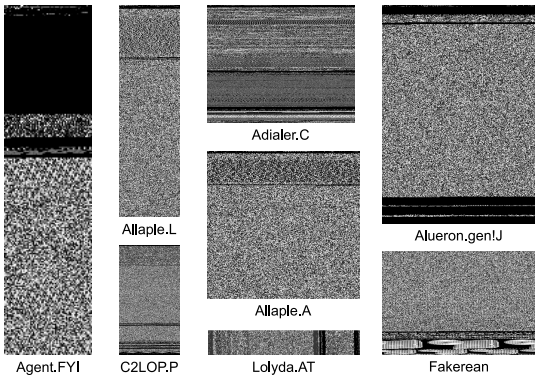

Fig. 1: Visual Representation (grayscale) of different malware classes from the `Malimg` dataset [1], [6].

In a seminal work, Nataraj *et al.* proposed using machine learning algorithms like Support Vector Machine (SVM), *k*-Nearest Neighbours (*k*-NN), etc. [1]. Gibert *et al.* suggested using Convolutional Neural Networks (CNN) [2] for malware classification using visual representation, while Kotha *et al.* in their work provided an ensemble model solution [3]. Al-Masri *et al.* suggested the use of Dual Convolution Neural Networks (DCNN), and therefore proposed Dual Convolutional Malware Networks (DCMN) for malware classification through images [4]. The effectiveness of several CNNs for malware image classification tasks was compared in [5]. Though efficient learning mechanisms are evolving to address the challenge of identifying and classifying the malware images, most malware visual analysis databases (for example, `Malimg`, `Big-2015`, etc.) are poorly balanced, making the classification yet more challenging. For instance, in the `Malimg` dataset [6], the class "`Allaple.A`" has 2959 data samples, while the class "`Skintrim.N`" has only 80 samples [7]. Further, each mal-

A. Dutta, S.P. Nayak and R.S. Chakraborty are with the Dept. of Computer Science and Engineering, IIT Kharagpur, Kharagpur, West Bengal, India 721302. Email: anuragdutta@ieee.org, satyapnayak@kgpian.iitkgp.ac.in, rschakraborty@cse.iitkgp.ac.in.
R. Naskar is with the Dept. of Information Technology, IIEST Shibpur, Howrah, West Bengal, India 711103. Email: ruchira@it.iiests.ac.in.

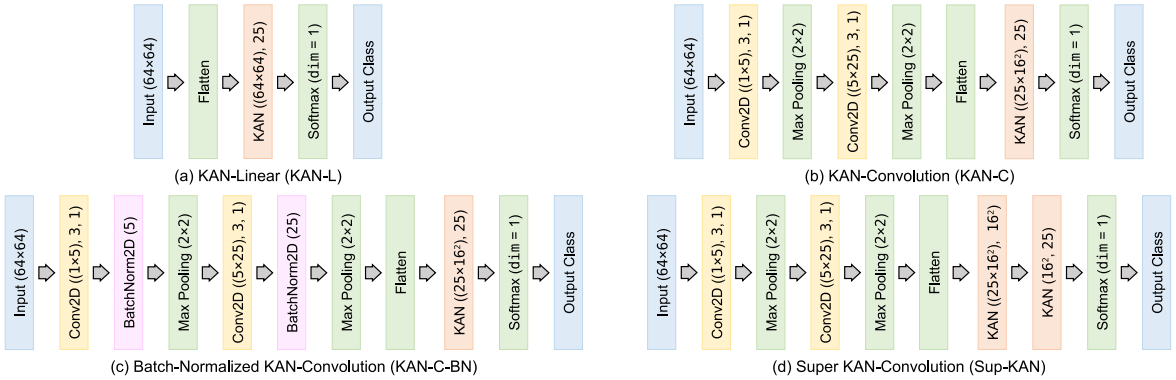

Fig. 2: Proposed architectures of (KOL-4-GEN) for the malware classification problem through visual representation.

ware image varies in its size and number of colour channels.

In this paper, we have applied a novel deep-learning architecture, namely the `Kolmogorov-Arnold Network` (KAN) [7], to perform accurate classification of malware binaries. The Kolmogorov-Arnold Network is unique, because rather than having *fixed activation functions* for nodal computations as Traditional multi-layered Perceptrons (MLPs) consist of a collection of *trainable activation functions* (preferably `B-Splines`) on the edges of the neural network [7]. **To the best of our knowledge, this is the first reported work on the application of KAN in malware forensics.** To mitigate the problem of *Dataset Imbalance* (if applicable) during the training phase, we employ a Generative Adversarial Network (GAN) to augment the training set with more synthetically generated samples for better balance. This enhancement offers better classification accuracy during *Validation*. We have developed and implemented four novel multi-modal KAN-based deep learning models, collectively named **KOL-4-GEN** for malware classification. Through extensive experimentation, we have compared their effectiveness vis-a-vis each other, against existing S-O-T-A techniques, applied on three open-source malware databases, `Malimg` [1], [6], `Malevis` [5], [8], and `Virus-MNIST` [9], [10]. Our experiments demonstrate that all four techniques of KOL-4-GEN outperforms all other existing schemes, while one of the four constituent techniques of KOL-4-GEN with two KAN layers (which we call "Super-KAN"), being the best in terms of validation accuracy, over both datasets.

II. PROPOSED METHODOLOGY: KOL-4-GEN

KOL-4-GEN, rather than a single proposed model, is a set of four deep-learning-based models inheriting KAN architecture, designed for multi-class malware classification. The models involve stacked computational layers, followed by a terminal output with `Softmax` function. Alongside, several well-known computational layers, like `convolution` (two-dimensional), `flattening`, `max pooling`, etc. are present; however, the novelty of the proposed models lie in the KAN layer.

A. The Kolmogorov-Arnold Network (KAN)

For any supervised learning problem with a training set, $\mathcal{T} \leftarrow \{\mathbf{x}_i, y_i\}$, where $\{\mathbf{x}_i\}$ are the feature vectors and $\{y_i\}$ are the respective targets, the objective is to find a function f such that, $f(\mathbf{x}) = y$. As per the *Kolmogorov-Arnold Representation Theorem* [11], to find the function $f(\cdot)$, it is necessary to compute the respective uni-variate, continuous functions Φ_i, and $\phi_{i,j}$ such that, for $\mathcal{N} \in \mathbb{Z}^+$, $f(\mathbf{x}) = \sum_{i=1}^{2\cdot\mathcal{N}+1} \Phi_i \left(\sum_{j=1}^{\mathcal{N}} \phi_{i,j}(\mathbf{x}_j) \right)$. Considering an input to the KAN layer of dimension n_{input}, and an output from the KAN layer of dimension n_{output}, the KAN layer can be interpreted as $\text{KAN}(\mathbf{x}) = \mathbf{\Phi} \cdot \mathbf{x}$, where $\mathbf{\Phi}$ is a matrix of learnable B-Splines as in Eqn.(1):

$$\mathbf{\Phi} = \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,n_{\text{output}}} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,n_{\text{output}}} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n_{\text{input}},1} & \phi_{n_{\text{input}},2} & \cdots & \phi_{n_{\text{input}},n_{\text{output}}} \end{bmatrix} \tag{1}$$

with, $\phi(x) = \omega_b \cdot \left(\frac{x}{1+e^{-x}}\right) + \omega_s \cdot \text{spline}(x)$, $\text{spline}(x) = \sum_i \lambda_i \cdot \text{B}_i(x)$, and $\lambda_i, \omega_b, \omega_s$ are trainable parameters that allow one to provide better control and interpretability.

B. The Four Classifier Models of KOL-4-GEN

The four classifier model architectures constituting KOL-4-GEN have been depicted pictorially in Fig. 2.

1) The **KAN-Linear** model is a raw implementation of the KAN layer and it simply passes a randomly selected input image patch of size 64×64 after flattening it. The output maps $(64 \times 64 \times x)$ inputs to y outputs, where x is the number of input channels, and y is the number of classes.
2) The **KAN-Convolution** model is a combination of a KAN layer with the classical 2-D convolution Layer. The first convolution layer considers an input with x channels, a kernel of size 3×3, and five output channels. The second convolutional layer considers an input with 5 channels, a kernel of size 3×3, and 25 output channels. They are followed by a two-dimensional `max pooling` layer, with a kernel size of 2×2. Finally, it is flattened and is passed through a KAN layer, mapping $(16 \times 16 \times 25)$ inputs to y outputs.

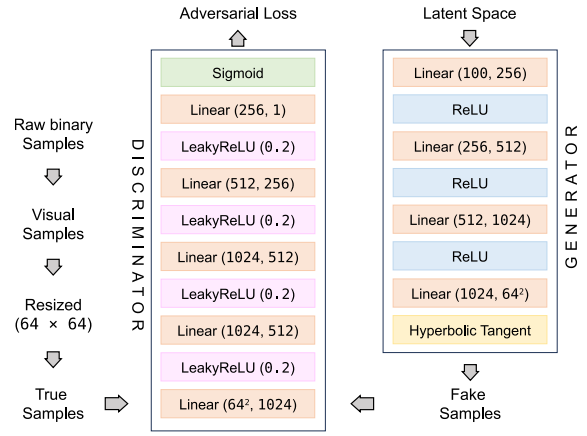

Fig. 3: The GAN architecture used to solve the dataset-imbalance problem.

3) The **Batch-Normalized KAN-Convolution** model is similar to the KAN-Convolution model, the only difference being an additional learnable `batch normalization` layer after each of the two 2-D convolutional layers. In our implementation, the number of output channels of the first normalization layer is parameterizable. (e.g. we have five output channels from the first convolution layer. Similarly, the second normalization layer has twenty-five out channels.
4) The best-performing model developed by us is **Super KAN-Convolution**. Following the dual convolution as in KAN-Convolution, the bits are passed through two consecutive layers of KAN – the first layer maps $(16 \times 16 \times 25)$ inputs to (16×16) outputs, and the adjacent second KAN layer maps selected (16×16)-sized inputs (which are output by the first KAN layer) to y outputs.

C. Solving the Dataset Imbalance Problem

To overcome the issue of dataset imbalance, KOL-4-GEN employs a Naïve Generative Adversarial Network (GAN) model (Fig. 3) with a latent space dimension of 100, and executed over 10 epochs. The GAN model was optimized iteratively using the `Adam Optimizer`, with a `learning rate` set to 2×10^{-4}; `Binary Cross-Entropy` (BCE) loss was employed for both the `Generator` and the `Discriminator`. Suppose, the raw dataset consists of a total of n images, and the GAN model augmented the dataset with a total of m additional synthetically generated images, i.e., $\left(\frac{m}{y} \text{ per class}\right)$. Considering an 80-20 training-test split, $0.8 \times (n+m)$ data points were used for training the models, and $0.2 \times n$ data points only from the raw dataset were used for validation, thus avoiding any possible data leakage and mitigating the imbalance problem.

III. EXPERIMENTAL RESULTS

A. Experimental Setup

The deep learning models under KOL-4-GEN (Section II) were implemented in Python using `PyTorch (v. 2.4)` as a back-end, and were evaluated on three (two imbalanced and one balanced) datasets. All experiments were carried out on a system with 16 GiB main memory, `Intel(R) Core(TM) i7-1065G7 @1.30 GHz` processor and an `NVIDIA GeForce MX330` Graphics Processing Unit with 2 GiB in-built memory. In our experimental evaluation, we considered three visual malware databases, (a) `Malimg`, an imbalanced gray-scale dataset with 25 malware classes and a total of 9339 multidimensional data points. [1], [6], (b) `Malevis`, a balanced RGB dataset with 25 malware classes and 1 non-malware class, with a total of 9100 data points (300 $\times$ 300 each). [5], [8], and (c) `Virus-MNIST`, an imbalanced miniature gray-scale dataset with 9 malware classes and 1 non-malware class with a total of 51880 data points (28 $\times$ 28 each) [9], [10]. Note that we were also inspired to use the datasets because of the contrasting natures of the dataset:

1) *Multi-modal computations*: `Malimg`, and `Virus-MNIST` consists of grayscale images, while `Malevis` of RGB Images.
2) *Multidimensional computations*: `Malevis` dataset consists of fixed 300 $\times$ 300 images, but `Malimg` contains images of variable dimensions. Additionally, the `Virus-MNIST` dataset consists of miniature images of 28 $\times$ 28 pixels each, which would be difficult for models to draw decision boundaries on, thus evaluating the models' robustness.

B. Evaluation

We have made datasets, code, and (due to page limit constraints) confusion matrix plots available at our project website [17]. Since this work is related to multi-class classification, common performance metrics such as Accuracy, Precision, Recall, and F1-Score were computed. Additionally, we also report the Time to Train (TTTr), Time to Test (TTTe) (i.e. inference time), and Model Size (in kilobytes) for different classifiers. Table I and Table II present the evaluation results and comparison with S-O-T-A. The following models were considered as baseline (mentioned along with their respective publication venues in chronological order by their publication years): Fine-tuned Convolutional Neural Network (IMCFN) [15], Depthwise Efficient Attention Module and DenseNet (DEAMD) [14], Multiscale Feature Fusion CNNs (MFFC) [13], Fine-tuned MobileNet (MNet_{FT}) [12], Custom CNNs (CNN_{CX}) [3], and Dual Convolutional Malware Network (DCMN) [4]. **Quantitatively, the proposed models outperformed the S-O-T-A, by an overall $\approx 2.3197\%$ improvement compared to the baseline models.** For satisfactory accuracy (better than the S-O-T-A, but lesser than the maximal achievable), one of the lightweight variants – **KAN-Linear** or **KAN-Convolution** may be chosen, while if computational cost is not a constraint, **Super KAN-Convolution** is desirable.

The relatively high training time may seem concerning. however, please note that this is an one-time effort. Once the classification model has been built, it can be deployed on commercially-available efficient hardware modules which are suitable for resource-constrained platforms, e.g. `NVIDIA Jetson Nano` or `Coral Dev Board`. To elucidate, the

TABLE I: Comparison of performance of different models for the `Malimg` dataset.

Metrics	Existing state-of-the-art Models						KOL-4-GEN Models (no augmentation)				KOL-4-GEN Models (with augmentation)			
	DCMN [4] (2024)	CNN_{CX} [3] (2024)	$MNet_{FT}$ [12] (2024)	MFFC [13] (2021)	DEAMD [14] (2021)	IMCFN [15] (2020)	KAN-L	KAN-C	KAN-C-BN	Sup-KAN	KAN-L*	KAN-C*	KAN-C-BN*	Sup-KAN*
							(proposed)				(proposed)			
Accuracy	0.9844	0.9867	0.9626	0.9872	0.9851	0.9827	0.9856	0.9898	0.9856	**0.9936**	0.9877	0.9877	0.9791	0.9898
Precision	0.9581	0.9859	0.9683	0.9886	0.9693	0.9825	0.9631	0.9767	0.9611	**0.9843**	0.9724	0.9674	0.9584	0.9754
Recall	0.9582	0.9903	0.9576	0.9872	0.9668	0.9819	0.9482	0.9737	0.9613	**0.9795**	0.9611	0.9680	0.9455	0.9710
F1-Score	0.9581	0.9912	0.9629	0.9873	0.9670	0.9820	0.9556	0.9752	0.9612	**0.9819**	0.9667	0.9677	0.9519	0.9732
TTTr (s)	517.62	893.65	1125.52	1984.04	1427.21	1037.38	1145.04	1433.31	1871.74	2068.95	1364.81	3662.86	3659.97	6972.44
TTTe (s)	4.28	3.56	4.83	5.34	4.34	3.79	4.57	4.71	5.36	6.32	4.90	10.73	11.02	12.43
Size (kB)	10253	23436	38170	43764	40133	30827	6275	9809	9812	96827	6275	9809	9812	96827

TABLE II: Comparison of performance of different models for the `Malevis`, and `Virus-MNIST` dataset.

Metrics	`Malevis` Dataset (balanced)					`Virus-MNIST` Dataset (imbalanced)								
	KOL-4-GEN Models (no augmentation)				S-O-T-A (2021)	KOL-4-GEN Models (no augmentation)				KOL-4-GEN Models (with augmentation)				S-O-T-A (2021)
	KAN-L	KAN-C	KAN-C-BN	Sup-KAN	ECOC [16]	KAN-L	KAN-C	KAN-C-BN	Sup-KAN	KAN-L*	KAN-C*	KAN-C-BN*	Sup-KAN*	LGBM [9]
Accuracy	0.8890	0.9429	0.9286	**0.9544**	0.9391	0.8943	0.9161	0.8995	0.9185	0.8965	0.8979	0.9016	**0.9212**	0.8800
Precision	0.9082	0.9431	0.9328	**0.9566**	0.9975	0.8646	0.8849	0.8835	0.8914	0.8617	0.8689	0.8638	**0.8950**	0.8850
Recall	0.8750	0.9405	0.9266	**0.9542**	0.9483	0.8519	0.8775	0.8554	0.8748	0.8550	0.8698	0.8712	**0.8970**	0.8670
F1-Score	0.8913	0.9418	0.9297	**0.9554**	0.9451	0.8577	0.8807	0.8678	0.8748	0.8580	0.8688	0.8642	**0.8931**	0.8758
TTTr (s)	885.68	1117.15	953.22	1600.79	1786.52	1698.08	4493.87	5339.25	7214.86	2222.56	5136.86	6572.44	9442.74	2217.15
TTTe (s)	3.87	5.61	5.59	7.09	6.38	9.15	10.07	10.18	12.05	10.22	14.09	14.15	14.87	10.07
Size (kB)	9539	10184	10187	96842	167856	2675	4184	4187	96602	2675	4184	4187	96602	2364

TABLE III: Implications of different computational layers for the Classification Process.

Layers (Particular to Classification)	Validation Accuracy (%)	Percentage Improvement	TTTr (sec.)
`Linear` (× 1)	95.04%	below baseline	728.94
`Linear` + `Convolution`	95.61%	below baseline	801.46
`Convolution` (× 2) **[baseline]** [1]	95.72%	baseline	1048.76
`KAN Linear` (× 1)	98.56%	≈ 2.9667%	1145.04
`KAN Linear` + `Convolution`	98.98%	≈ 3.4058%	1433.31
`KAN Linear` (× 2) + `Convolution` (× 2)	99.36%	≈ 3.8028%	2068.95
`KAN Linear` (× 3) + `Convolution` (× 3)	99.25%	≈ 3.6887%	5329.14

metric TTTe ("Time to Test") was added, similar practice was followed in the work by Wang et al. [13], where "prediction time" was considered as a decisive metric to choose a suitable input shape of malware binaries.

It was observed that the accuracy achieved by the baseline (with no KAN layer) fell short compared to the models with KAN layers, though up to a threshold of two consecutive KAN layers. Introducing additional KAN layers makes the model suffer possibly from overfitting, resulting in poorer accuracy than its counterparts. Table III gives the respective accuracies, training time (TTTr), and percentage improvement in prediction accuracies for different computational layers.

IV. CONCLUSIONS

We have developed KOL-4-GEN, a set of four novel deep-learning models for malware classification from their visual representations. Our techniques incorporate the KAN architecture with trainable activation functions and enhance it with GAN-generated synthetic images to resolve the issue of unbalanced datasets. The models were evaluated on multi-modal datasets, outperforming S-O-T-A methods in terms of common classification metrics.

REFERENCES

[1] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware Images: Visualization and Automatic Classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, 2011, pp. 1–7.

[2] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using Convolutional Neural Networks for Classification of Malware represented as Images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, pp. 15–28, 2019.

[3] E. Kotha, L. H. Palivela, and A. Begum, "Enhanced Malware Image Classification through Ensemble model," in *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. IEEE, 2024, pp. 1421–1425.

[4] B. Al-Masri, N. Bakir, A. El-Zaart, and K. Samrouth, "Dual Convolutional Malware Network (DCMN): An Image-Based Malware Classification Using Dual Convolutional Neural Networks," *Electronics*, vol. 13, no. 18, p. 3607, 2024.

[5] A. S. Bozkir, A. O. Cankaya, and M. Aydos, "Utilization and Comparision of Convolutional Neural Networks in Malware Recognition," in *2019 27th signal processing and communications applications conference (SIU)*. IEEE, 2019, pp. 1–4.

[6] L. Nataraj *et al.*, "Malimg Dataset," https://paperswithcode.com/dataset/malimg, 2011, Accessed: September 2024.

[7] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," *arXiv preprint arXiv:2404.19756*, 2024.

[8] A. S. Bozkir *et al.*, "MaleVis: A Dataset for Vision Based Malware Recognition," https://web.cs.hacettepe.edu.tr/~selman/malevis/, 2019, Accessed: September 2024.

[9] E. Larsen, K. MacVittie, and J. Lilly, "Virus-MNIST: Machine Learning baseline calculations for Image Classification," *arXiv preprint arXiv:2111.02375*, 2021.

[10] E. Larsen, *et al.*, "Virus-MNIST Dataset," https://www.kaggle.com/datasets/datamunge/virusmnist, 2021, Accessed: November 2024.

[11] J. Schmidt-Hieber, "The Kolmogorov–Arnold representation theorem revisited," *Neural networks*, vol. 137, pp. 119–126, 2021.

[12] M. P. Salas and P. L. de Geus, "Deep Learning applied to Imbalanced Malware Datasets Classification," *Journal of Internet Services and Applications*, vol. 15, no. 1, pp. 342–359, 2024.

[13] S. Wang, J. Wang, Y. Song, and S. Li, "Malicious code variant identification based on Multiscale Feature Fusion CNNs," *Computational Intelligence and Neuroscience*, vol. 2021, no. 1, p. 1070586, 2021.

[14] C. Wang, Z. Zhao, F. Wang, and Q. Li, "A Novel malware detection and family classification scheme for IoT based on DEAM and DenseNet," *Security and Communication Networks*, vol. 2021, no. 1, p. 6658842, 2021.

[15] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based Malware Classification using Fine-tuned Convolutional Neural Network architecture," *Computer Networks*, vol. 171, p. 107138, 2020.

[16] W. K. Wong, F. H. Juwono, and C. Apriono, "Vision-Based Malware Detection: A Transfer Learning approach using Optimal ECOC-SVM configuration," *IEEE Access*, vol. 9, pp. 159 262–159 270, 2021.

[17] A. Dutta *et al.*, "KOL-4-GEN Datasets and Codes," https://github.com/Anurag-Dutta/KOL-4-GEN, 2024, Accessed: September 2024.