Shallow Deep Learning using Space-filling Curves for Malware Classification

David Long and Stephen O'Shaughnessy
Technical University Dublin, Ireland
davielong86@gmail.com
Stephen.OShaughnessy@tudublin.ie

Abstract: The incidents of malware attacks are continually increasing at a rapid rate, thanks to the lucrative potential in schemes such as ransomware, credential stealing Trojans and cryptominers. Their explosive growth is compounded by the ease with which variants can be created from original strains. As a result, anti-virus organisations are struggling to keep up, with some reporting upwards of 14 million samples processed per month. These sheer volumes have caused a shift towards machine learning and artificial intelligence in an effort to alleviate the manual burden of analysis and classification. This research presents a novel framework for the classification of malware into distinct family classes through computer vision and deep learning. In the proposed framework, malware binaries are represented in an abstract form as images mapped through mathematical constructs known as space-filling curves. Convolutional neural networks were constructed and applied to the malware images to build predictive models for classification. The models were optimised using an auto-tuning function for the hyper parameters, which included Bayesian Optimisation, Random search and HyperBand, providing an exhaustive search on the hyper parameters. On a training dataset of 13k malware samples from 23 distinct families, the models yielded an average score of 95% for precision, recall and f1-score. The final deep learning model was validated for robustness against a dataset of more recent variants, comprising 12,816 samples from 16 malware families, returning classification scores of 95%, 86% and 90% for precision, recall and f1-score. The final model was demonstrated to outperform a similar benchmark model considerably. The results show the potential of the deep learning framework as a viable solution to the classification of malware, without the need for manually intensive feature generation or invasive processing techniques.

Keywords: Malware image classification, Deep Learning, Computer Vision, Space-filling curves, H-curve

1. Introduction

The continuing global upsurge in malware attacks can be greatly attributed to the potential profits to be gained from schemes such as ransomware and credential stealing Trojans. Anti-virus companies, such as AV-Test, report over 450,000 new malicious programs processed daily (AV-Test, 2021). Analysis on this scale is not manually feasible and so analysts and researchers constantly strive to produce more scalable solutions. More and more, researchers are utilising machine learning as a means of alleviating the manual analysis burden of malware classification. Typically, features are collected and then are passed to classification algorithms to train models for prediction, so feature generation, i.e., the extraction and subsequent choice of features, is crucial to the success of the model. Broadly speaking, features can be extracted in two ways: statically (from non-running binaries) or dynamically (from binaries in execution). Common static features are in the form of frequency or sequence data, such as n-gram or PE file header analyses (Khalilian et al., 2018; Kolter and Maloof, 2004). Dynamic features are extracted by executing malware and recording the behaviour by monitoring its interactions with the operating system, for example through API call sequences and graph representations (Fukushima et al., 2010). However, these approaches are not without their limitations. Frequency or sequence-based methods are context dependent, so any word or frequency contexts outside of a given range are not recognised; dynamic methods are prone to producing large volumes of false positives as much of the behaviours recorded are common to benign applications also.

Computer vision and image processing techniques have been presented as an alternative to the more typical approaches to malware classification (Wagner et al., 2015). Under these approaches, binary code is mapped or transformed into image formats, such that the images retain characteristics that can be extracted as discriminant features for identification and classification. The benefit of image representations is that the conversion process is less invasive than other methods of analysis such as those discussed previously, which simplifies feature generation. This paper presents a novel method of malware family classification that combines computer vision and deep learning through convolutional neural networks (CNNs). Malware binaries are mapped to a 2-dimensional image format through space-filling curves (SFCs). SFCs are mathematical constructs whose ranges contain the entire 2-dimensional unit square i.e., the 2-dimensional unit square represents an image of n x n pixels. SFCs traverse through every pixel point of a regular spatial region, such that the spatial locality of the data

is preserved. This is significant, as it enables the internal structure of the original malware binaries to be retained and represented in the resulting images, which can then be used for classification purposes.

In summary, this research makes the following contributions:

- *A novel approach to malware image classification through the application of Deep Learning & Computer Vision to malware Space-filling curve images.*
- *Comparative analysis with benchmark previous work to determine viability of proposed method.*
- *A pre-processed labelled malware dataset consisting of approximately 26k samples in space-filling curve image format.*

2. Related work

The visualisation of binary files as images was introduced in the work of Conti et al. (2008), whereby a method of reverse engineering binary files into visual images, dubbed "byteplots", was presented to enhance the capabilities of text-based hex editors. In this work, Conti et al. derived a visual taxonomy of binary file fragments using byteplots, to aid in the identification of common file formats. Conti et al. (2010) extended their previous work by applying machine learning to classify file types by their visual features. On a dataset of 14k samples comprising 14 file types, including machine code, encrypted data and plain text, the authors reported high classification scores, particularly from unencoded files, where accuracies of 98.7% and above were recorded. While the initial work presented by Conti et al. did not focus on malware classification, it demonstrated that the internal static structure of binary files could be represented as 2-dimensional images, which fostered further work in the application of byteplots to malware classification.

Previous works in image-based malware classification have focused predominantly on the byteplot representation for image-based malware classification. Nataraj et al. (2011) were the first to apply the byteplot mapping in this context. The authors derived feature vectors using GIST filters tuned to varying scales and orientations. They evaluated their method using a dataset, dubbed Malimg, comprising 9,548 samples from 25 malware families, reporting an average classification accuracy of 92%. Vasan et al. (2020) presented a method of transfer learning using the established pretrained VGG16 and ResNet-50 architectures with multiclass SVMs as classifiers, applied to the Malimg dataset for malware family classification. The authors reported an average accuracy score of 99.5% on their ensemble method, which out-performed 14 other similar methods. Le et al. (2018) used 6 multi-layered convolutional neural networks on binary files from the Kaggle malware dataset (Kaggle, 2015) that were converted to grayscale images. Results on the data showed 98.8% classification accuracy.

In contrast, limited works exist on SFCs in the image-based malware classification domain. This paper extends the work of O'Shaughnessy (2019), where the efficacy of space-filling curves was evaluated as a means of representing malware variants for classification. Three SFC image datasets were produced by mapping 9,235 32-bit executable malware samples from 28 distinct families to images using Z-order, Hilbert and Gray-code curve traversals. Features extracted via Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG) and Gabor filters were used to train Random Forest, K-Nearest Neighbour and Support Vector Machine classifiers. The best results were obtained from the KNN-HOG model trained on the Z-order dataset, with precision, recall and accuracy metric scores of 94.5%, 87.1% and 91.6% respectively. A comparative assessment with the method presented by Nataraj et al. (2011), showed the KNN-HOG Z-order SFC model outperformed the GIST byteplot method against previously unseen samples.

This research addresses several of the limitations associated with previous works. First, the framework utilises SFC images to classify malware. SFCs have previously demonstrated better classification performance compared to the predominant byteplot method (O'Shaughnessy, 2019). Second, the malware datasets used in this research were compiled from recent repositories which are representative of current malware variants, compared with the Malimg and Kaggle datasets, which were compiled in 2011 and 2015 respectively. Finally, this research explores the use of deep learning applied to SFC images for malware classification, which has not been documented previously.

3. Methodology

The research presented in this paper provides a study of deep learning and space-filling curves as a means of classifying malware into their respective family classes. The method of the study comprises three distinct parts:

data gathering & pre-processing, data conversion and classification, illustrated in Figure 1. It should be noted that this research is only concerned with the classification of malware into distinct family classes and as such no benign samples were used for this research.

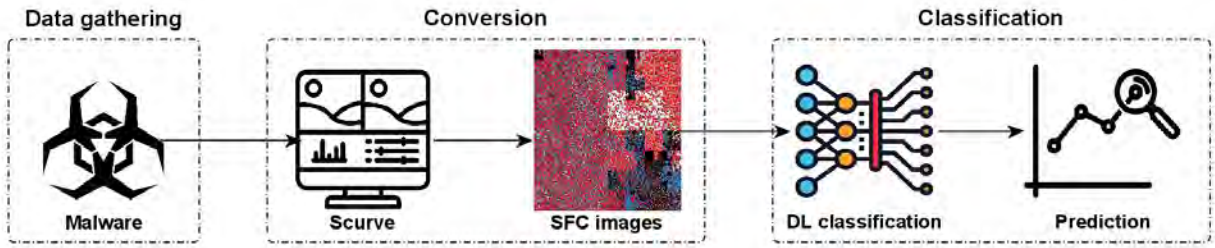


Figure 1: Deep learning malware classification framework architecture

3.1 Data Gathering and Pre-processing

This phase involved gathering and preparing the malware samples for conversion and classification. The samples were collected from the VirusTotal (VT) academic collection (VirusTotal, 2021). The data used to train the deep learning models consisted of 13,091 malware samples obtained during the period January 2018 - December 2019. A further dataset was compiled from January 2020 to July 2021 for validating the models. The validation set was compiled from a more recent timeline to demonstrate the robustness of the models on newer generation variants. It should be noted that due to the lack of samples from some families in more recent repositories, the validation dataset comprised 12,816 samples from 16 out of the 23 families in the training set. Since the samples were not labelled, it was a necessary pre-processing step to cluster the data into their respective family classes. The labels for the samples were obtained through the AVClass labeller tool (Malicia, 2016). This tool parses the VT reports that accompany each sample and predict a class label according to the majority anti-virus detections. This enabled the fast processing of the samples into their distinct family classes.

3.2 Malware Conversion

Space filling curves were chosen as the traversal mapping to convert malware binaries to image format. Space-filling curves are a mathematical construct whose range contains the entire 2-dimensional unit square or more generally an n-dimensional unit hypercube. In this case, the 2-dimensional unit square represents an image of n x n pixels. SFC's trace a continuous curve through every unit square, i.e., pixel in the image.

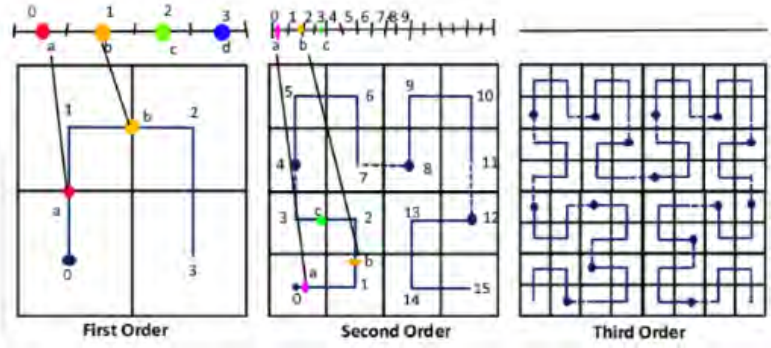


Figure 2: 1-dimension to 2-dimension mapping through Hilbert curve traversal. (a) first order, (b) second order and (c) third order.

Figure 2 depicts the mapping of a 1-dimensional line (representing the sequential malware binary code) to a 2-dimensional curve representation, in this case the Hilbert curve, in first, second and third order iterations. It is evident that closely located points (a and b) map to similar positions on the SFC image. The significance of the SFC traversal mapping in this context is the locality preservation of the data in the resulting images, i.e., similar regions in the malware binary code are grouped together by clusters or regions of distinct colours or textures. In this regard, it has been demonstrated previously by O'Shaughnessy (2019) that malware variants, who by nature share considerable sections of code, map to similar textures within the SFC images. This then allows for the images to be used as representations of the malware for classification purposes.

For the purposes of this research, scurve, a Python visualisation library, was used to convert the malware binaries into SFC image format (Cortesi, 2012). The scurve library uses a colour-coding scheme, based on data type, that maps the malware binary data to the resulting SFC images. The colour scheme classifies bytes into the following categories: black for 0x00, blue for ASCII text, red for high-value bytes, and white for 0xff. These mappings give the SFC images distinctive texture patterns. The images in Figure 3 show samples from three malware families: Dridex, Emotet and Sfone. If the data type from the malware binary is distinct, it is displayed as non-overlapping sections of a single colour. For example, Figure 3 (a) contains distinct regions of black, denoting binary zeros whereas Figure 3 (b) contains regions of distinct blue, denoting ASCII characters. The rest of the images comprise a mixture of multiple data types, which give the resulting image distinct texture regions. The SFC traversal method chosen for this research was the H-curve implementation, which has been demonstrated previously to possess the most favourable locality preservation properties over other curves. Furthermore, the Hilbert, Z-order and Gray-code curves have previously been explored, whereas there currently exists no work utilising the H-curve traversal.

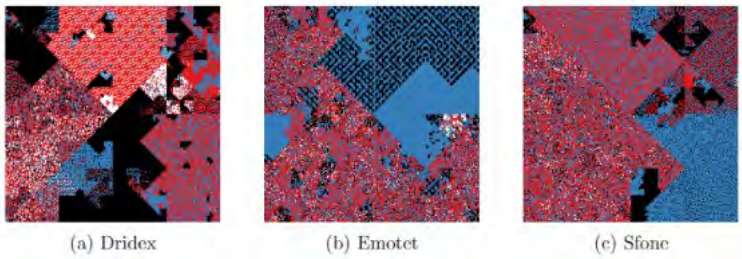


Figure 3: SFC samples in H-curve format

3.3 Classification

The aim of classification in this context is to order the malware samples into their taxonomic groups or family classes, based on shared textural similarities. Convolutional Neural Networks (CNN's) were employed in this research as they had not been explored in similar works previously. This form of classification has been demonstrated as a viable method for image recognition in many different areas. The main classification model was implemented using python Tensorflow (Tensor flow, 2021) and Keras (Keras Team, 2021).

3.3.1 Hyperparameter Auto-tuning

The DL models were constructed through an extensive hyperparameter tuning process using the three Keras auto-tune functions namely, Random Search (RS), Bayesian Optimisation (BO) and Hyperband (HB). RS operates by selecting a random set of HPs and testing the models in an iterative approach, selecting a random set of HPs for each iteration. BO treats the HP set as a regression problem which seeks to find the minimal loss in the minimal amount of processing time. HB provides an extensive search by iteratively sampling combinations of HPs over a defined number of epochs. It selects the optimum set based on the results, which is then used to conduct the final training.

The models trained consisted of a basic model structure as did the base model used while identifying the best HP set for the models. The range for the number of layers that the models were trained on was set between 2 and 10. This was done to keep complexity as low as possible to ensure the training could be carried out within the confines of the available computational power. It was found that 10 layers made it infeasible to execute the entire training process and so it was set as the upper bound for the number of layers in each model. Rectified Linear Unit (ReLu) and Softmax activation functions were utilised in both the model training and the final model. The ReLU function was applied at the convolutional and dense layers of the model, and Softmax was applied to the final output layer to classify each malware sample input into its respective family class.

3.3.2 Model Architecture

In the pursuit of an architecture that is in line with the novel approach identified, the option of a predefined architecture was discarded. The reason being that, while there are several pre-defined CNN architectures that have proven effective in image classification, such as Res-Net50 or VGG16, their complexity and subsequent burden on computational power made them infeasible for the proposed architecture. Instead, a simplified custom architecture was devised, where high classification output performance was achievable, while remaining within the feasible limits of computational power. Furthermore, since the SFC images were represented in a

pixelated form, there was less need for the intensive filtering used on the images as is typical in pre-defined architectures. The models that were constructed for this research required just enough complexity to be effective for classification, while remaining simplistic enough to execute within the limits of the analysis machine, referred to as the "Goldie-locks-zone" (Fort and Scherlis, 2018).

Through the implementation of the models with the auto-tuning process, a network architecture was identified for the final model. This structure was determined through an iterative process of adding a convolutional layer with corresponding max-pooling and dropout layers until the optimum results were achieved. In the final model, discussed in Section 4.2, the outputs of each convolutional layer are connected to a dense layer and a flattening layer, then to the output layer with a softmax activation to classify each input into their respective classes of malware. This approach classifies the one-dimensional representation of the malware binary files using two-dimensional texture patterns of each SFC image as features. The proposed CNN architecture, illustrated in Figure 4, consists of 5 convolutional layers, 7 dropout layers, 3 dense layers and a flattening layer.

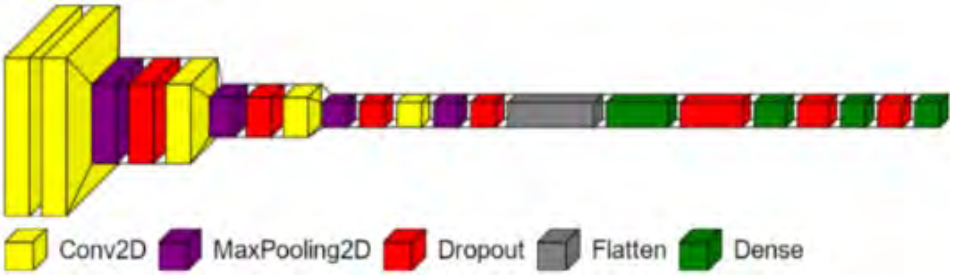


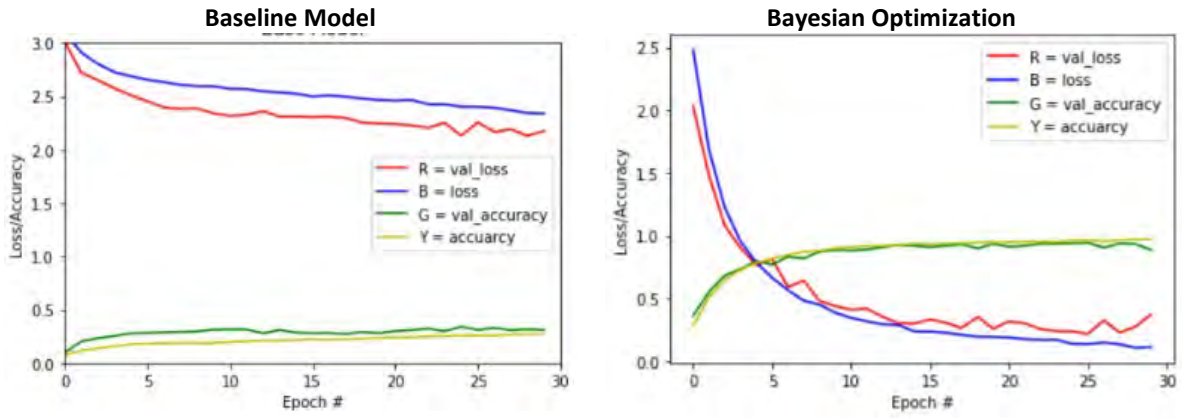
Figure 4: Final model architecture.

4. Analysis & Results

This section discusses the experimental methods devised to optimise the CNN architecture in terms of classification performance and time complexity. Optimal hyper parameters were determined through the auto-tuning functionality of the Keras-Tuner library. The metrics used to determine classification performance were primarily accuracy, precision, recall and F1-score. Plots were used to provide a graphical view of the model's loss and accuracy over the range of epochs used.

4.1 Model Hyperparameter Tuning

A default configuration was devised where no tuning took place, in order to obtain a baseline accuracy to gauge the performance of the auto-tuners. This baseline HP architecture was determined from previous research in image-based classification, such as Le et al. (2018). The baseline and Keras auto-tuners were then applied to the dataset. The loss and accuracy plotted over time (epochs) for each configuration is shown in Figure 5. The x-axis represents the number of epochs, and the y-axis represents loss and accuracy for the training and validation stages. From Figure 5, the baseline training and validation loss are convergent, meaning that the model did not overfit the data. However, the training and validation accuracies are low, with none above 55%. The auto-tuners performed considerably better, as expected. Both the BO and RS compared similarly for training and validation phases, also producing convergent training and validation loss, with accuracy scores above 90% from epoch 10 and saturating at approximately epoch 20.



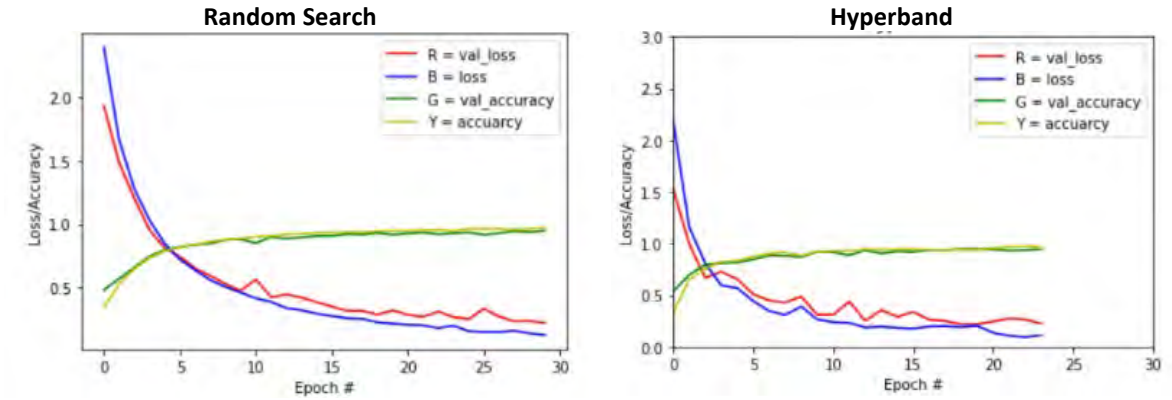


Figure 5: Tuning loss/accuracy performance

The classification performance of the baseline and auto-tuners was next investigated. Table 1 shows the performance and time complexity for each method. The baseline model returned performance metrics of 61%, 48% and 46% for precision, recall and f1-score respectively. This is in line with the outputs from the loss and accuracy plot discussed in Figure 5. The BO tuner performed less favourably than both the RS and HB tuners, with precision recall and F1-scores of 92%, 89% and 89%, which constituted a loss of between 3% and 5%. Furthermore, BO tuning was the slowest of the tuners, constituting almost one second per sample in the training phase. Both RS and HB yielded the same results, with precision, recall and F1-scores of 95%. However, the RS tuner HP set was chosen for the final model as training was almost twice as fast as HB.

Table 1: Model HP training classification performance and time complexity

Tuner Type	Precision %	Recall	F1 Score %	TPS (train)	TPS (val)
Random Search	95	95	95	0.451	0.040
Bayesian Opt.	92	89	89	0.937	0.119
Hyperband	95	95	95	0.823	0.118
Baseline (no tuning)	61	48	46	0.697	0.040
TPS = Time per sample					

4.2 Final Optimised Model

The final model was constructed from the optimised HP RS set, coupled with an architecture comprising 5 convolutional layers, 7 dropout layers, 3 dense layers and a flattening layer, obtained through an iterative process, as discussed in section 3.2.2. The number of epochs chosen for the final model was 20, as accuracy stabilised beyond this point. The recorded increased validation accuracy between epoch 18 and 30 was negligible and so an early stopping parameter was set when training the model. When it reached a given value, the gains from further training were deemed negligible and the model stopped training. Further research into the optimum number of epochs would be irrelevant in this case as the dataset is not large enough to warrant a more extensive training process.

4.2.1 Model Validation

If the final deep learning model is to be considered robust, it must be impervious against various perturbations in malware variants as they evolve or mutate over time. Therefore, the final model was evaluated using a validation dataset that was collected from a later timeline, as discussed in Section 3.1. The loss and accuracy plot for the final model on the validation data is shown in Figure 6. The model achieved low training and validation loss values, indicating that the classification error rate was low. Additionally, both loss curves are convergent, which means the model did not overfit the data, so generalised well to the previously unseen data. The model also returned a high classification accuracy for both training and validation, reaching ~95% after epoch 5 and stabilised for the remainder of the experiment. These outputs show that the model was robust in predicting newer generations of family variants.

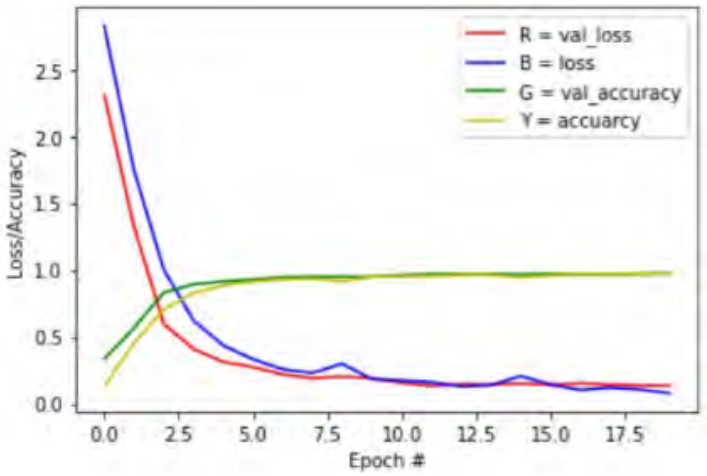


Figure 6: Final model loss and accuracy performance

Table 2 shows the classification performance of the final model for each malware family class. While the model performed well overall, there were some families that achieved low classification rates. For example, the model performed poorly on the Emotet and Scar families. In both cases, precision and accuracy scores were low. Low precision in this case means that when the model predicted a specific class of malware, in many cases it was incorrect. Low recall means there is a high incidence where the model incorrectly predicted other malware families as a particular class. However, the dataset compiled for this research is imbalanced, due to the lack of availability of some family samples. From Table 2 the sample count, both Emotet and Scar contained low numbers of samples (44 and 91 respectively), in which case there may have been insufficient features to train the models on, leading to poor classification performance.

The class imbalance issues were compounded by the evolving nature of the malware. To provide illustrative examples, Figure 7 shows Emotet samples taken from the training (a-c) and validation (d-f) datasets. It is evident that the texture patterns are different across all example SFC images. Emotet has undergone many changes and evolutions since its inception in 2014, where Emotet's developers have changed its functionality and increased obfuscation features in order to evade detection.

Table 2: Performance of the final model on the validation dataset

Malware family	Precision %	Recall %	F1 Score %	Sample Count	% Of sample count
Allaple	81	97	88	264	1.92
Berbew	99	71	82	4146	35.35
Dinwod	98	96	97	911	7.1
Dorkbot	56	95	71	130	1
Emotet	7	55	13	44	0.34
Fsysna	88	72	79	188	1.47
Hematite	94	99	97	670	5.23
Oberal	33	97	49	101	0.79
Picsys	94	99	97	295	2.3
Salgorea	98	93	95	1235	9.64
Scar	16	19	17	91	0.71
Sfone	99	98	99	2719	21.21
Socks	79	100	88	30	0.23
Sytro	96	87	91	991	7.73
Vilsel	93	100	97	70	0.55
Vobfus	94	98	96	931	7.26
Weighted avg.	**95**	**86**	**90**		

This results in the code structures differing greatly as the variants evolve. This produces disparate SFC images as displayed in Figure 7 which in turn had a detrimental effect on the prediction capabilities of the models.

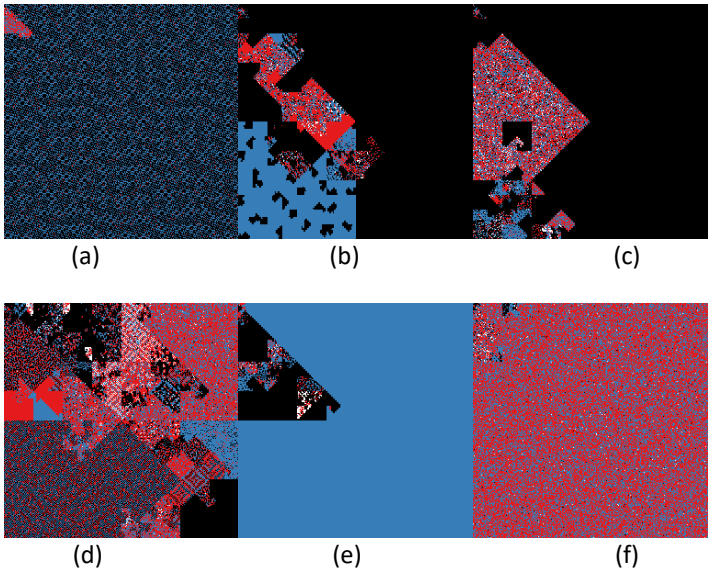


Figure 7: Emotet samples taken from the training (a-c) and validation (d-e) datasets

4.2.2 Time Complexity Optimisation

The initial conversion process produced SFC images of 256 x 256 pixels as this is the default setting in the scurve library. However, the average time complexity per sample to train the model was 8.8 seconds, which can be considered infeasible for large datasets. The optimisation of time complexity through smaller dimension images was thus considered. The size parameter in the scurve library was tested at various dimensions and it was found that training time was greatly reduced to 1.1s per sample, when the SFC images were converted to 32 x 32 pixels, resulting in an accuracy loss of just 1.1%. This allowed the training and subsequent validation testing to be conducted within a reasonable timeframe, which enhances the scalability of the proposed method. It should be noted that the final tuned model achieved moderately faster time complexities of 0.925s and 0.041s per sample for training and validation, respectively.

4.2.3 Comparison with previous benchmark method

To test the feasibility of the proposed deep learning method for image-based malware classification, a comparative analysis was conducted with the previously proposed method by O'Shaughnessy (2019), discussed in Section 2. This method can be considered a benchmark as it was the first documented work using space-filling curves for malware classification. The best performing model, KNN-HOG, was chosen for comparative purposes. The KNN-HOG model was trained and evaluated on the same H-curve SFC training and validations sets as the proposed deep learning method. Table 3 shows the performance results for training and validation for both methods. From the table, KNN-HOG outperformed the proposed method marginally in the training phase, with precision, recall and accuracy scores of 97%. However, in the validation set, the deep learning model performed considerably better in all three performance metrics. The results show that the proposed deep learning model is more robust than the benchmark model, which demonstrates its feasibility as a viable method for image-based malware classification.

Table 3: Performance comparison for KNN-HOG and DL models on training and validation data

	Training			Validation		
Model	**Precision %**	**Recall %**	**F1-score %**	**Precision %**	**Recall %**	**F1-score %**
KNN-HOG	97	97	97	83	63	63
DL	95	95	95	**95**	**86**	**90**

5. Discussion

The application of deep learning to classify SFC malware representations had not previously been explored, which prompted the main motivation of the study. Furthermore, previous research focused on gathering features for classification through invasive or complex malware analyses methods. The framework presented in this paper provides a method for malware classification that negates the need for intensive feature generation or invasive analysis techniques. By utilising convolutional neural networks, feature extraction and selection is not necessary as the models identify different levels of image representations by learning the basic textures in

the first layers and evolving to learn features of the image in the deeper layers. The CNN architecture can be considered shallow in that it consists of just 5 convolutional layers, which simplifies the model's complexity for more efficient computation, enhancing scalability for larger datasets. Additionally, the final model returned high prediction rates on a validation set of variants from a more recent timeline, demonstrating its robustness against evolved malware family variants. Finally, in order to evaluate the viability of the proposed method, a comparative analysis was conducted with the final model and the benchmark method by O'Shaughnessy (2019). The results show the final model significantly outperformed the benchmark model on the validation dataset, proving its viability.

6. Conclusions and Further Work

The main aim of this research was to produce a malware classification solution that was robust and scalable to overcome current limitations. The findings presented here are significant in that they demonstrate the feasibility of proposed deep learning framework as a robust, scalable method for image-based malware classification. Limitations exist, including the size of the dataset and the format of the malware. CNNs have been shown to perform best on larger datasets. There is scope to increase the malware dataset to improve the model's performance. The malware format chosen to investigate was the Windows Portable Executable format, which represents the format used in the majority of malware produced. Again, research could be expanded to include other formats such as malicious PDF or Word Macros.

References

AV-Test, 2021. Malware Statistics & Trends Report | AV-TEST [WWW Document]. URL /en/statistics/malware/ (accessed 2.18.21).

Conti, G., Bratus, S., Shubina, A., Lichtenberg, A., Ragsdale, R., Perez-Alemany, R., Sangster, B., Supan, M., 2010a. A Visual Study of Primitive Binary Fragment Types 17.

Conti, G., Bratus, S., Shubina, A., Sangster, B., Ragsdale, R., Supan, M., Lichtenberg, A., Perez-Alemany, R., 2010b. Automated mapping of large binary objects using primitive fragment type classification. Digital Investigation 7, S3–S12. https://doi.org/10.1016/j.diin.2010.05.002

Conti, G., Dean, E., Sinda, M., Sangster, B., 2008. Visual Reverse Engineering of Binary and Data Files, in: Goodall, J.R., Conti, G., Ma, K.-L. (Eds.), Visualization for Computer Security, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 1–17. https://doi.org/10.1007/978-3-540-85933-8_1

Cortesi, A., 2021. cortesi/scurve, https://github.com/cortesi/scurve.

Fort, S., Scherlis, A., 2018. The Goldilocks zone: Towards better understanding of neural network loss landscapes. arXiv:1807.02581 [cs, stat].

Fukushima, Y., Sakai, A., Hori, Y., Sakurai, K., 2010. A behavior based malware detection scheme for avoiding false positive, in: 2010 6th IEEE Workshop on Secure Network Protocols. Presented at the 2010 6th IEEE Workshop on Secure Network Protocols, pp. 79–84. https://doi.org/10.1109/NPSEC.2010.5634444

Institute, M.L.@ I.S., 2021. AVClass and AVClass2.

Kaggel, 2015. Microsoft Malware Classification Challenge (BIG 2015) [WWW Document]. URL https://kaggle.com/c/malware-classification (accessed 3.3.21).

Keras Team, 2021. Keras documentation: Developer guides [WWW Document]. URL https://keras.io/guides/ (accessed 6.21.21).

Khalilian, A., Nourazar, A., Vahidi-Asl, M., Haghighi, H., 2018. G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families. Expert Systems with Applications 112, 15–33. https://doi.org/10.1016/j.eswa.2018.06.012

Kolter, J.Z., Maloof, M.A., 2004. Learning to detect malicious executables in the wild, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04. Association for Computing Machinery, New York, NY, USA, pp. 470–478. https://doi.org/10.1145/1014052.1014105

Le, Q., 2018. Deep learning at the shallow end: Malware classification for non-domain experts. Digital Investigation 9.

Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011. Malware images: visualization and automatic classification, in: Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec '11. Presented at the the 8th International Symposium, ACM Press, Pittsburgh, Pennsylvania, pp. 1–7. https://doi.org/10.1145/2016904.2016908

Nataraj, Lakshmanan, Yegneswaran, V., Porras, P., Zhang, J., 2011. 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 - AISec '11. Presented at the the 4th ACM workshop, ACM Press, Chicago, Illinois, USA, p. 21. https://doi.org/10.1145/2046684.2046689

O'Shaughnessy, S., 2019. Image-based Malware Classification: A Space Filling Curve Approach, in: 2019 IEEE Symposium on Visualization for Cyber Security (VizSec). Presented at the 2019 IEEE Symposium on Visualization for Cyber Security (VizSec), IEEE, Vancouver, BC, Canada, pp. 1–10. https://doi.org/10.1109/VizSec48167.2019.9161583

Tensor flow, 2021. Libraries & extensions | TensorFlow [WWW Document]. URL https://www.tensorflow.org/resources/libraries-extensions (accessed 4.21.21).

Vasan, D., Alazab, M., Wassan, S., Safaei, B., Zheng, Q., 2020. Image-Based malware classification using ensemble of CNN architectures (IMCEC). Computers & Security 92, 101748. https://doi.org/10.1016/j.cose.2020.101748

VirusTotal, 2021. VirusTotal – Learning resources. URL https://www.virustotal.com/learn/ (accessed 8.28.21).

Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D.A., Aigner, W., 2015. A Survey of Visualization Systems for Malware Analysis. Presented at the Eurographics Conference on Visualization (EuroVis), pp. 105–125. https://doi.org/10.2312/eurovisstar.20151114