
11-785 Final Report

Jared Broyhill

Carnegie Mellon University
Pittsburgh, PA 15213
jbroyhil@andrew.cmu.edu

Jai Chauhan

Carnegie Mellon University
Pittsburgh, PA 15213
jchauhan@andrew.cmu.edu

Abstract

Traditional malware detection is based on signatures, which are byte sequences or hash values used to identify portions of malicious code. However, modern day malware is often polymorphic in that it can encrypt or obfuscate its code to evade detection by signature-based tools. Further, it takes significant human effort to classify and understand the operation of malware. This project addresses these challenges by implementing a Vision Transformer (ViT) model to classify malware images from the MalNet dataset, which is a repository of more than a million malware images labeled by type and family. Our model achieved a classification accuracy of 74.6% and demonstrated the ability to generate useful attention maps that highlight significant regions in malware code that led to its classification. These attention maps can benefit security researchers in identifying malware without manual analysis of binary executables. Additionally, we explore Graph Attention Networks (GAT) for examining function call graphs (FCGs), achieving an accuracy of 81.9% in distinguishing malware types based on program structure. The results demonstrate an overall effectiveness of attention-based approaches for explainable malware classification.

1 Introduction

1.1 Motivation

Malware has historically and continues to pose a significant risk to computer systems worldwide. The exploitation of these systems can have effects that range from inconvenience to catastrophe. To safeguard against such negative occurrences, traditional malware utilizes signature-based detection methods that rely on known byte sequences or hash values to identify malicious code. However, modern malware employs various techniques to evade these methods:

- Polymorphic malware can encrypt or obfuscate its own code, making signature matching increasingly ineffective [1].
- New malware variants emerge regularly and outpace the updates of signature databases
- The complexity and nuance involved in understanding malware behavior often requires significant effort and expertise on the part of human security practitioners. [2]

The increasing complexity of malware has led to a greater need for more effective and adaptable detection and classification methods. Deep learning has emerged as an alternative with promising prospects, given its ability to automatically learn the relevant features of a given predictive problem from raw data.

The approach of representing malware as images has been particularly influential in the cross section between deep learning and information security since it was pioneered [3]. These visual representations typically capture structural patterns that persist even when the malware itself is designed to evade traditional detection mechanisms. Malware image representations eliminates the

need for manual feature engineering and transforms the problem of malware detection into an image classification problem.

In contrast to image-based static analysis, malware function call graphs (FCGs) are a useful aid for dynamic analysis to detect suspicious activity on an already infected machine. FCGs represent programs as networks of function calls. Nodes in the graphs represent functions, and edges represent caller-callee relationships. Advancements in graph neural networks have made classification of even the most complex FCGs a reality.

Explaining model classification decisions is crucial for researchers. Attention-based heat maps overlaid on malware images or FCGs can help indicate particularly malicious code segments or functions.

A more secure world—a safer world—requires more robust solutions for security. A deep learning approach, utilizing cutting-edge advancements in predictive capabilities such as transformer architectures, presents a worthwhile area of investigation for this purpose.

Our primary objectives for this project were to build a vision transformer (ViT) model to classify Android malware images from the MalNet dataset. Then, generate attention maps to identify regions of suspicious code that influence classification decisions. Additionally, build a Graph Attention Network (GAT) to classify malware Functional Call Graphs (FCGs) from the MalNet dataset. Finally, compare the performance of our models with benchmarks from previous research in the domain.

2 Literature Review and Background

Malware analysis as a research domain within the field of deep learning has attracted significant interest in recent years. Several key contributions have shaped our approach.

The primary motivation for our project is the paper by Jeonggeun Jo et al [4]. In this paper, they used a Vision Transformer (ViT) model to classify images from AndroZoo (one of original data sources for Malnet) and also used the “attention rollout” mechanism (originally proposed by [5]) to build the attention maps. The paper went so far as to convert the relevant elements from the attention map into readable code. This enabled them to identify and extract the highlighted portions of code that contributed to the classification decision, effectively creating an explainable AI system for malware analysis. Their research also demonstrated that transformers could match and exceed the performance of convolutional neural networks for the same task. This paper was not released publicly, so we have reverse engineered their model’s transformer architecture and attention rollout mechanism.

The Malnet dataset itself provides contribution to the subject of malware visualization. Frietas et al. [6] provides over 1.2 million malware images across 47 types and 696 families. The Malnet dataset is one of the largest and most complex cybersecurity image databases publicly available. The authors established important baseline performance metrics using various CNN architectures, such as ResNet. With ResNet 18%, they achieved a macro-F1 score of 47%. Considering that this work used the same dataset as our research, it is included within the baseline consideration.

The Malnet graph dataset [7] appears to be understudied, providing us the opportunity to conduct novel analysis. The dataset authors experimented with a variety of graph models including graph convolutional networks (GCN), and Graph Sample and Aggregate Networks (GraphSAGE). There does not appear to be much research on the use of attention mechanisms on the graph dataset.

3 Methodology

3.1 ViT Model Description

Our model implements a ViT architecture (originally introduced by Dosovitskiy et al. [8]) designed for malware image classification. It processes images by splitting them into fixed patches. Linear embedding is performed for each patch, and then position embeddings are added, followed by the results being passed through transformer encoder blocks. This model is directly inspired by the ViT approach of Jo et al., who implemented a very similar model for their purposes, but did not supply any source code.

The general architecture is as follows: image preparation is first, the MalNet Dataset’s DEX files are resized to 224x224x3 and normalized following ImageNet statistics. Next, 2D image data is converted into 16x16 non-overlapping patches, transformed into sequences (patching). Following that, a class token is prepended to the created sequence. A positional encoder accounts for spatial information to patches with sine and cosine functions. Next, a transformer encoder with 12 layers is positioned. It utilizes 8-headed multi-head attention to compute and normalize the score with softmax to create a probability distribution, and attention probability weights for weighted context. A feed-forward network with 1024 dimensions then introduces non-linearity and capacity into the model. Finally, there is a classification head that produces class predictions and output logits for each malware class.

A figure that offers a visual depiction of the model in a process diagram is provided for the reader:

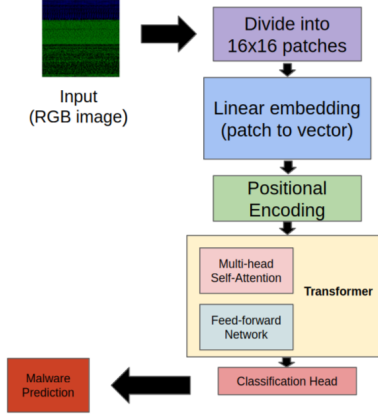


Figure 1: ViT architecture for malware classification

The table below details the parameters and hyperparameters of our Vit Model.

Table 1: ViT Model architecture		
Total Parameters	Hidden layers	Parameters/hyper-parameters
12,844,847	$12 \times \{\text{Transformer Block}\}$	Number of heads (8) Embedding dimension (256) Patch size (16) Feed-Forward dimension (1024) Embedding dropout (0.3) Attention dropout (0.4) Feed-Forward dropout (0.4) Learning Rate (0.0001) Batch Size (32) Number of patches (196)

3.1.1 Attention Rollout

Originally introduced by Abnar et al. [5], Attention Rollout is a mechanism to provide explainability, specifically for ViT models. In a multi-layer transformer architecture with self-attention, information from patches becomes mixed. This makes it difficult to understand the influence of images patches on the final classification decision.

The below equations are taken from the original paper. Since the transformer architecture has a residual connection around each multi-headed attention block, the output of any block can be described as:

$$V_{l+1} = V_l + W_{attn} V_l \quad (1)$$

This can be simplified to:

$$V_{l+1} = (W_{attn} + I)V_l \quad (2)$$

Note that adding the identity matrix to the attention weight matrix would violate the property that each row of the attention matrix should sum to 1. Hence this term is "renormalized":

$$A = (0.5 * W_{attn} + 0.5 * I) \quad (3)$$

Next is the rollout phase. While iterating over each layer, i, calculate:

$$A'_i = A_i A'_{i-1} \quad (4)$$

This resource [9] provided some inspiration for the implementation. First, our model averages the weights across heads in each layer for a given image during inference. The weights of each layer are then stacked to build W_{attn} and the attention roll is performed as described above. Once complete, we simply take the first row of the final rollout matrix, which corresponds to the weights of the class token. The final weight matrix is reshaped, and overlaid over the final image.

3.2 Graph Attention Network Model Architecture

Traditional Graph Convolution Networks (GCNs) involve 'message passing' layers in which nodes exchange their node feature vectors with its neighbors. Graph Attention Networks originally proposed by Velickovi et al. [10] add an attention mechanism to these layers where each node learns an attention coefficient for each of its neighbors.

The MalNet graph dataset contains edgelist for each FCGs, without any defined node features. For this reason we used Local Degree Profile (LDP) as node features. Originally proposed by Adamczyk et al. [11], LDP summarizes degree information about a node and its neighbors. As defined by [11], DN is the multi set of degrees of neighboring nodes.

Then the following statistics make up a node's LDP: 1.degree(v), 2.min(DN), 3.max(DN), 4.mean(DN) 5.std(DN). This results in a node feature vector of length 5. The figure below summarizes our GAT architecture.

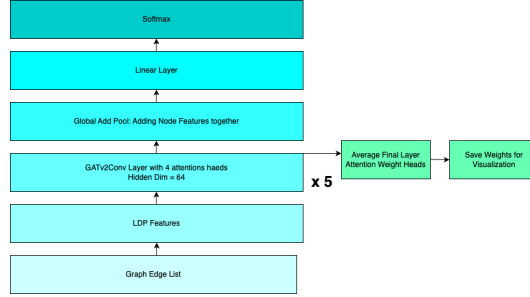


Figure 2: GAT architecture

Since we are performing graph-level classification, a global add pool layer is required to sum the node features before being passed to the linear layer. Note that attention weights from the final GATv2Conv layer are saved for visualization. The table below summarizes the parameters and hyperparameters of our GAT model:

Table 2: GAT Training Hyper-Parameters

Total Parameters	Learning Rate	Weight Decay	Dropout	Batch Size	Epochs
2,672,005	0.00005	0.05	0.1	32	100

Based on our ablations, the model appeared to be quite sensitive to learning rate.

3.3 Image - Dataset

The Malnet dataset can be accessed here: <https://www.mal-net.org/>. The dataset classifies Android malware into 47 types (one of which is benign) and 696 families. The dataset is imbalanced with some families having far more examples than others. The dataset contains roughly 1.2 million images, and is about 80GB in size.

In the MalNet dataset, Android DEX (Dalvik Executable) files are converted to images are created by interpreting their binary content as pixel values. Separate sections of the DEX file—such as the header, string table, and class definitions—often create distinct visual patterns that neural networks have been demonstrated to be able to recognize.

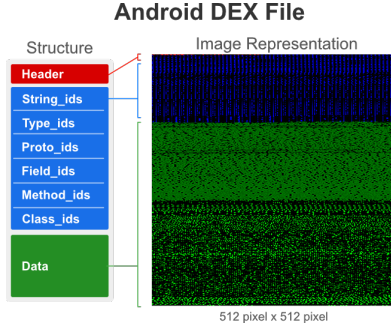


Figure 3: Android DEX file structure, composed of three major components—(1) header, (2) ids, and (3) data. Right: binary image representation of the DEX file. (Reprinted from Freitas et al., 2022)

Due to existing computational restraints of the computing environment in which we trained the model, we were unable to train the model on the full size of the dataset. The Malnet authors provided a 10% subset (around 120,000 images) that matched the class distribution of the full dataset. This subset claimed to follow a standard 70%, 10%, and 20% train, val, test split. Likely due to an error by the dataset authors, the training dataset was later discovered to be the smallest of these three subsets. This goes against best practices in deep learning and perhaps impacted our results.

3.4 Graph - Dataset

The Malnet dataset also contains more than 1.2 million FCGs, resulting in more than 300GB of graph data. Due to computational constraints, our work focused on the "Malnet-Graphs-Tiny" subset. This subset includes roughly 5000 FCGs spanning 5 different malware types. Unlike the image dataset, the graph subset has a roughly uniform distribution across classes.

3.5 Evaluation Metric

For both the image and graph models, our key evaluation metrics are accuracy, macro-F1 scores, and weighted F1-scores. Accuracy provides an overall measure of correct classifications, but may be misleading due to the imbalanced nature of our dataset where some classes are more common than others. Macro-F1 score gives equal weight to each class regardless of frequency and thus provides insight into model performance across all malware types, including rare ones. Finally, the weighted-F1 score accounts for class frequency and provides a more realistic assessment of expected real-world performance where some types of malware are more common. These metrics are optimal due to their combined ability to comprehensively represent our results.

When it comes to malware detection (malware vs benign) we aimed to meet and ideally exceed the baseline metric values. In this situation false positives are more acceptable than false negatives due to the security concerns presented by the latter.

3.6 Loss Function

We implemented a standard cross-entropy loss function without class weighting for both our ViT and GAT models. For a classification problem with C classes, the cross-entropy loss is defined as:

$$\mathcal{L}_{CE} = - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(p_{i,c}) \quad (5)$$

where:

- N is the number of samples
- $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for sample i
- $p_{i,c}$ is the predicted probability that sample i belongs to class c

This approach treats all classes equally during training, regardless of frequency. Comparisons with a version of our model that utilized weighted cross entropy loss to mitigate class imbalance is included in "Results."

4 Baseline and Extension

4.1 Image Classification Baseline

The results from Jo et al. [4] are provided below, where a ViT was applied to malware classification on the AndroZoo dataset, which is smaller and significantly less complex than the MalNet dataset.

Model	Accuracy	F1 Score	Precision	Recall	F1 Score	Precision	Recall
Weighted Avg				Macro Avg			
our proposed model	0.8027	0.7743	0.7775	0.8027	0.3113	0.5219	0.2509
ResNet18	0.8005	0.7737	0.7762	0.8005	0.2748	0.4800	0.2207
ResNet50	0.7843	0.7514	0.7528	0.7843	0.2230	0.3767	0.1803
DenseNet121	0.8047	0.7770	0.7764	0.8047	0.2397	0.3831	0.1983
MobileNetV2	0.7701	0.7258	0.7333	0.7701	0.1776	0.3803	0.1381

Figure 4: Jo et al. baseline results

Jo et al.’s approach is methodically aligned with our research objectives of accurate classification and explainable results with a ViT architecture, hence its selection as the primary baseline. They achieved strong performance metrics and established that their transformer-based model could effectively capture visual patterns that distinguish malware types, hence why they served as the primary blueprint for our approach.

Provided below are the various metrics from the Malnet paper [6] using various standard CNN architectures.

Model	Param	GFlop	Binary			Type			Family		
			F1	P	R	F1	P	R	F1	P	R
RN18	12M	1.8	.86	.89	.84	.47	.56	.42	.45	.54	.42
RN50	26M	3.9	.85	.91	.81	.48	.57	.44	.47	.54	.44
RN101	45M	7.6	.86	.88	.84	.48	.59	.44	.47	.54	.44
DN121	7.9M	2.9	.86	.90	.83	.47	.56	.43	.46	.53	.44
DN169	14M	3.4	.86	.89	.84	.48	.57	.43	.46	.55	.43
MN2(x.5)	1.9M	0.1	.86	.89	.83	.46	.55	.42	.45	.53	.42
MN2(x1)	3.5M	0.3	.85	.89	.83	.45	.53	.42	.44	.53	.41

Table 2: We evaluate the performance of 3 architectures—ResNet, DenseNet and MobileNetV2—on its macro-F1, macro-precision, and macro-recall. We conduct the remaining experiments using the ResNet18 model as it provides a strong balance between performance and efficiency.

Figure 5: Freitas et al. baseline results

The metrics are important points of comparison for our ViT implementation; they serve as performance measurements on the specific dataset we are using. Notably, both papers utilized a ResNet18-based model, hence why we utilized it for our baseline replication.

4.2 Graph Classification Baseline

The Malnet paper authors utilized a GCN model with LDP features against the Malnet-Graphs-Tiny dataset achieving an accuracy of .81. This serves as an appropriate baseline as we will be able to accurately measure the benefits gained by our GAT model. .

4.3 Extensions

Our work is novel in the sense that we applied a ViT based model to MalNet, with the goal of exceeding the existing metrics of Freitas et al. that were the product of a ResNet18-based CNN approach to malware image classification, and Jo et al., who also utilized a ResNet18-based approach, but for a ViT.

Our GAT model represents an extension to the Malnet graph dataset, with the potential to exceed the baseline accuracy while also providing a notion of explainability.

5 Results and Analysis

5.1 Image Model Results

Due to computational and time restrictions, we were unable to train our model as much or as long as we would have liked. As such, we trained it for 50 epochs. It's compared to the baseline replication, trained to 48 epochs, and the full baseline, trained to 100 epochs.

The data gathered during the training and validation, are shown below.

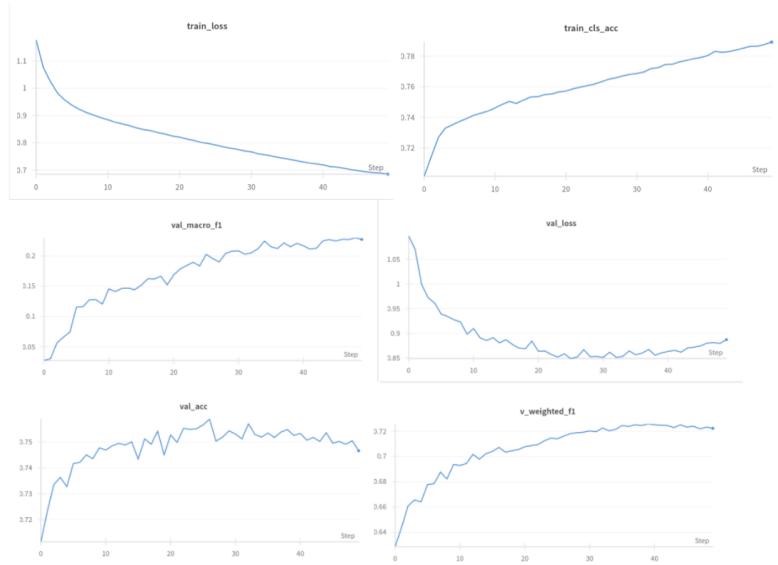


Figure 6: ViT Training and Validation Data

The table below summarizes the results of our ViT model, the replicated baseline and the original baseline. Our replicated baseline was comparable, but ultimately missing some elements of the original approach.

Table 3: ViT results

Model	Test Acc	Macro-F1	Weighted-F1
ViT	0.746	0.214	0.721
Original ResNet18 Baseline	0.80	0.275	0.774
Replicated ResNet18 Baseline	0.78	0.337	0.762

The following figure is a confusion matrix depicting the most common 15 classes (malware types):

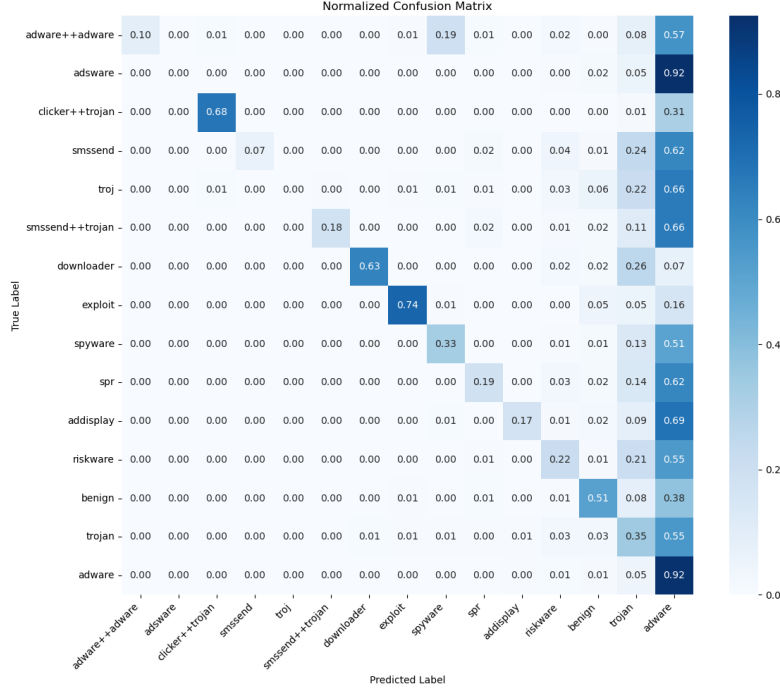


Figure 7: Confusion matrix generated by our ViT.

5.1.1 Discussion

Our results reveal that our ViT is robust. From early epochs, it had and continued to maintain a strong baseline accuracy that required minimal tuning. Dominant patterns were likely learned quickly given these results. While performance of our model is poorer than the replicated and genuine baseline, it is comparable, and appears to effectively capture most class patterns. The training-validation gap stayed under five points as well throughout training, stabilizing at around a three-to-four points difference, suggesting that the model is not overfitting. It also suggests that our ViT is generalizing well to unseen data.

The confusion matrix demonstrates a high per-class recall on dominant malware types within the dataset (e.g., "adware" and "exploit.") This suggests a high capability at correctly identifying dominant malware types. Minority types, meanwhile, are frequently absorbed into dominant bucket, suggesting that our ViT is not as apt at capturing less frequent malware types accurately. Additionally, misclassifications between related classes on the matrix underscore the fact that the model is likely capturing behavioral similarity; it is likely detecting the products of actual behavior and code similarities before the malware was converted into RGB images.

We believe that there are numerous reasons our ViT was unable to perform as well as the baseline. The first is existing computational restraints: due to issues present in our computing environment, as well as the massive breadth of the MalNet Dataset and the overhead it had, we were unable to train as much we would have liked (i.e., 100 epochs). Additionally, the MalNet dataset itself is not

only highly imbalanced, the splits offered by the dataset authors do not adhere to best practices: the train dataset is smaller than the validation and testing sets. This likely contributed to the results not exceeding the baseline.

Furthermore, conventional methods to mitigate class imbalance issues in the MalNet dataset did not prove successful. We attempted weighted cross entropy loss as a mechanism to improve the model’s accuracy, but it instead severely reduced it. While the training accuracy eventually stabilized, macro-F1, weighted-F1, and validation loss never did. The unstable validation data is provided below:

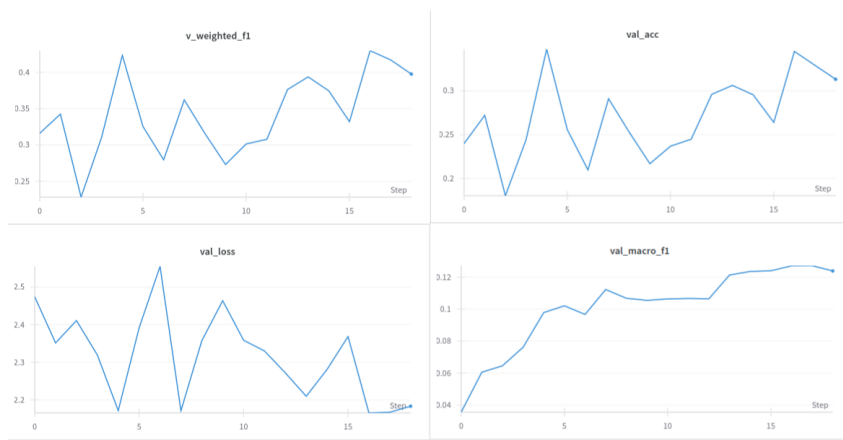


Figure 8: ViT with weighted cross-entropy loss validation data

Despite challenges, the attention maps produced as a product of the ViT demonstrate significant utility for malware analysis.

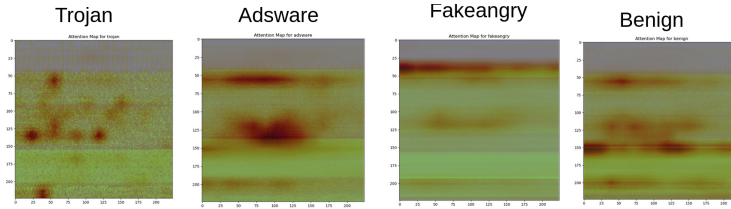


Figure 9: Attention maps for "trojan," "adware," "fakeangry" malware types, and a benign attention map

It can be seen that the areas of attention highlighted differ in the three examples of malware attention maps. Considering how ViT classification functions, these areas of attention are very likely to correspond to malicious segments of code in the executable used to create the images. However, the benign attention map has a greater dispersion of attention, suggesting more components of its code played a role in its classification. This provides unique insight into the otherwise opaque nature of identifying malware through deep learning models—it appears that while smaller segments of code can easily distinguish malware, a greater extent of code may be necessary for the model to classify a subject as benign.

5.2 Graph Model Results

The below table summarizes the results from the re-created baseline and our GAT model.

Table 4: Graph Model Results		
Model	Accuracy	Macro-F1
GCN (Baseline)	0.847	0.8519
GAT (our model)	0.819	0.8164

As shown we were able to roughly replicate the reported baseline results. Our GAT model performed slightly worse than the baseline, but we suspect that the model may scale better to a larger dataset. The below figure shows two FCG samples in which the edge color and thickness determined by the attention weights from the final convolutional layer

Note that the FCGs in the dataset typically have tens of thousands of nodes. The smaller examples below were hand-picked for easier visualization.

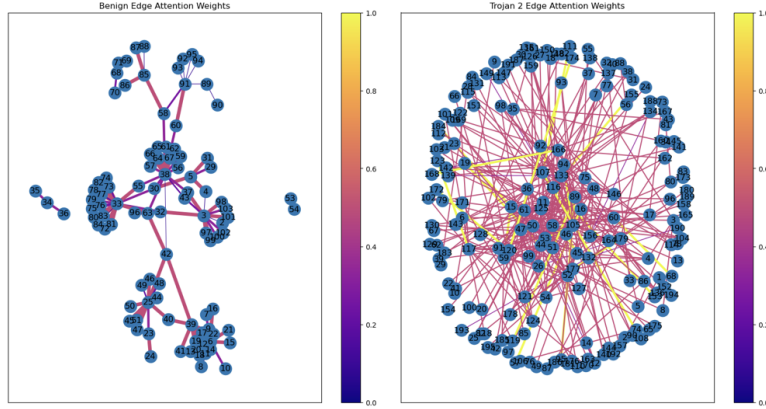


Figure 10: Edge-Weighted FCGs

As with the attention maps, different malware types exhibit distinct patterns of edge colors/thickness. The benign FCGs appears to have more variety in edge attention weights, while the trojan FCG has distinct high attention edges. This supports our aforementioned suggestion from the image attention maps, offering worthwhile evidence as to how neural networks classify malware.

6 Conclusion and Future Work

Our ViT model achieved promising results but was limited by computational constraints and class imbalance in the image dataset. A weighted cross-entropy loss function was not effective in addressing the imbalance. This suggests that the dataset should be augmented with more minority class examples. Further, Generative Adversarial Networks (GANs) could possibly be used to generate synthetic images. Also when trained on the full dataset, the ViT will be exposed to more minority class examples and may see an improvement in F1-Scores.

Our GAT model demonstrated promising results on the graph subset. Further experiments should be conducted on the full graph dataset. Future work can also explore graph transformers, a promising architecture.

The key success of our project was generating both attention maps through attention rollout and edge-weighted FCGs. Future work could focus on open source tools to extract relevant code from such visuals. This could be an invaluable contribution to the information security community, who often contend with the threat of zero-day vulnerabilities (vulnerabilities that have yet to be publicly exploited). The capability of identifying novel exploits sooner offers significant protection for computer systems and the people who rely on them.

In conclusion, we have shown that attention mechanisms have the potential to enhance malware graph classification and image models while also providing explainability, which can greatly benefit security researchers.

7 Administrative Details

7.1 References

- 1 Chaikovskiy, Maksym, et al. "Comprehensive approach to the detection and analysis of polymorphic malware." CEUR Workshop Proceedings. Vol. 3736. 2024.
- 2 Saha, Bikash, Nanda Rani, and Sandeep Kumar Shukla. "MaLAWare: Automating the Comprehension of Malicious Software Behaviours using Large Language Models (LLMs)." arXiv preprint arXiv:2504.01145 (2025).
- 3 Nataraj, Lakshmanan, et al. "Malware images: visualization and automatic classification." Proceedings of the 8th international symposium on visualization for cyber security. 2011.
- 4 Jo, Jeongeun, Jaeik Cho, and Jongsub Moon. "A malware detection and extraction method for the related information using the ViT attention mechanism on android operating system." Applied Sciences 13.11 (2023): 6839.
- 5 Abnar, Samira, and Willem Zuidema. "Quantifying attention flow in transformers." arXiv preprint arXiv:2005.00928 (2020).
- 6 Freitas, Scott, Rahul Duggal, and Duen Horng Chau. "MalNet: A large-scale image database of malicious software." Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2022.
- 7 Freitas, Scott, et al. "A large-scale database for graph representation learning." arXiv preprint arXiv:2011.07682 (2020).
- 8 Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).
- 9 jeonsworld. "ViT Pytorch" GitHub, 2020, https://github.com/jeonsworld/ViT-pytorch/blob/main/visualize_attention_map.ipynb Accessed 30 Apr. 2025.
- 10 Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).
- 11 Adamczyk, Jakub, and Wojciech Czech. "Strengthening structural baselines for graph classification using Local Topological Profile." International Conference on Computational Science. Cham: Springer Nature Switzerland, 2023.

7.2 Relevant Wandb Runs

- ViT best: <https://wandb.ai/jbroyhil-cmu/Malware-Classification/runs/8e71ihs0/overview>
- ViT with weighted cross-entropy loss: <https://wandb.ai/jbroyhil-cmu/Malware-Classification/runs/7dsvzohn/overview>
- Replicated baseline: <https://wandb.ai/jbroyhil-cmu/Malware-Classification/runs/8mrj7anr/overview>
- GCN baseline: <https://wandb.ai/11-485-Group-HW/Malware-Classification/runs/gzle7dxy?nw=nwusergocheckmate>
- GAT best: <https://wandb.ai/11-485-Group-HW/Malware-Classification/runs/o928ohb6?nw=nwusergocheckmate>

7.3 Team Contributions

- Jared Broyhill - Initial ViT model architecture and creation, refining and engineering ViT, baseline replication, fine-tuning, research, writing, presenting
- Jai Chauhan - Attention map implementation, Graph Model development, report writing
- Overall work was split equally.

7.4 GitHub

https://github.com/jbroyhill/IDLMalwareImageClassification/blob/main/malware_project_v01.ipynb