A HYBRID CNN AND VISION TRANSFORMER MODEL FOR MALWARE CLASSIFICATION

Benat Froemming-Aldanondo, Christopher Hardwick, Ryan Hill & Vinh Nguyen
Department of Computer Science
University of Minnesota
Minneapolis, MN 55414, USA
{froem076,hardw050,hill1886,nguy4766}@umn.edu

ABSTRACT

In this work, we discuss classifying malware using deep learning image recognition techniques applied to executable files represented as images. Static classification methods are emphasized as they help detect malware before execution, reducing potential harm. This study explores two different multiclass classification problems: malware families with and without benign software. Traditional methods struggle with polymorphic malware and high false positives, whereas deep learning provides a promising alternative by capturing discriminative patterns directly from raw executable data. Using a hybrid CNN and ViT model, we achieved a higher accuracy on the VirusMNIST dataset than the original while maintaining high weighted precision, recall, and thus F1 scores for classifying different families of malware. Additionally, we applied transfer learning with ResNet and EfficientNet and achieved higher accuracy than the original when classifying malware families and beneware with EfficientNet-B0.

1 INTRODUCTION

1.1 THE PROBLEM

In this work, the overarching problem we will discuss is classifying malware using deep learning image recognition techniques applied to executable files represented as images. There are three methods that are generally used to classify malware: static methods that classify malware based on the executable before it is run, dynamic methods that classify malware while the program is running, and hybrid approaches that use a combination of both. We will be focusing on static methods because it is important to classify malware before it is run so it causes the least amount of harm. The challenge involves two key classification tasks: first, distinguishing between malicious and benign software (malware vs. beneware), and second, identifying specific malware families. The binary classification problem is critical for cybersecurity, as accurately detecting malicious executables before they execute can prevent system compromises. Traditional detection methods like hash-based signatures struggle with polymorphic malware that alters its code to evade detection, while heuristic approaches often generate excessive false positives. Deep learning offers a promising alternative by learning discriminative patterns directly from raw executable data. The multiclass classification problem is equally important, as correctly categorizing malware types (e.g., ransomware, trojans, or downloaders) helps analysts prioritize threats and understand attack methodologies during reverse-engineering (Maniriho et al., 2024).

1.2 THE INTRIGUING NATURE OF THE PROBLEM

Malware detection given a binary executable file typically uses heuristics and hash-based signature matching to make its classifications (Maniriho et al., 2024). What makes this problem particularly interesting is the counterintuitive effectiveness of treating binary executables as images for classification purposes. While the resulting grayscale representations of PE headers are not interpretable to human analysts, deep learning models have demonstrated surprising success in extracting meaningful patterns from this data.

Prior work has shown that convolutional neural networks can achieve over 80% accuracy in malware detection using these image representations, despite the models being originally designed for natural image recognition (Noever & Noever, 2021). This suggests that the spatial arrangement of bytes in executable headers contains latent structural signatures that deep learning can detect, even when traditional analysis methods might miss them. Interpretability is not a major goal, so the lack of it in deep learning isn't an issue. The approach also offers practical advantages, such as faster inference times compared to dynamic analysis methods and potential for real-time detection systems.

1.3 Previous Work

A recent literature review by Maniriho et al. shows the increase in deep learning and machine learning pieces in the past few years. Included works use deep learning methods that include, but are not limited to recurrent neural networks, deep neural networks, autoencoders, and feed-forward networks. Studies similar to the one we will perform classified executables based on extracted features including an executables assembly instructions and byte streams, with the most popular features being opcode and binary images constructed from the executable (Maniriho et al., 2024).

Previous research has established important benchmarks in this domain. The VirusMNIST dataset authors achieved 80% accuracy for malware detection and 87% accuracy for family classification using MobileNetV2 on their image-formatted PE headers (Noever & Noever, 2021). Other studies like MalNet demonstrated that larger datasets (1.2 million samples across 696 classes) could achieve 84% accuracy, suggesting that scale improves model performance (Freitas et al., 2022). However, VirusMNIST remains valuable as a more manageable benchmark for testing novel approaches, especially given computational constraints. Building upon these findings, researchers have explored various convolutional neural network architectures to enhance classification performance. For instance, an ensemble of CNN models demonstrated improved detection rates for both packed and unpacked malware samples, demonstrating the efficacy of combining multiple models to capture diverse feature representations (Vasan et al., 2020).

The integration of transformer-based models has also been investigated. A study comparing the performance of ResNet-152 and Vision Transformer architectures found that ViT models could effectively capture intricate patterns in grayscale malware images, offering a promising alternative to traditional CNN approaches (Ashawa et al., 2024).

The STAMINA project by Intel and Microsoft further validated the potential of image-based malware classification, achieving excellent results with custom CNN architectures, though their exact methods and dataset specifics were not fully disclosed, affecting the reproducibility of this method (Chen et al., 2020).

1.4 This Work

The primary goals of this work was to improve upon the existing 80% and 87% classification accuracy achieved on VirusMNIST for two different problems. We utilized the VirusMNIST dataset for model development and testing. We investigated the application of ViT architectures and developed hybrid architectures that combine CNNs and transformers to make use of the strengths of both approaches. In this hybrid model, CNN layers are used to extract low-level features before passing the data to a Transformer for global understanding. Additionally, we implemented transfer learning using ResNet-18 (He et al., 2015) and EfficientNet-B0 (Tan & Le, 2020) as base models to determine the effectiveness of our handcrafted model considering its additional design complexity with the simplicity of using existing models.

2 Methodology

2.1 VirusMNIST Exploratory Data Analysis

We performed a comprehensive exploratory data analysis of the VirusMNIST dataset. The dataset contains 48,422 training samples and 3,458 test samples, each consisting of 1,024 grayscale pixel values and a corresponding label ranging from 0 to 9. No missing values were found in either the training or test sets, so no imputation or data cleaning was necessary. Sample images from each class were visualized to provide an initial sense of the visual variability between malware types in Figure 1.

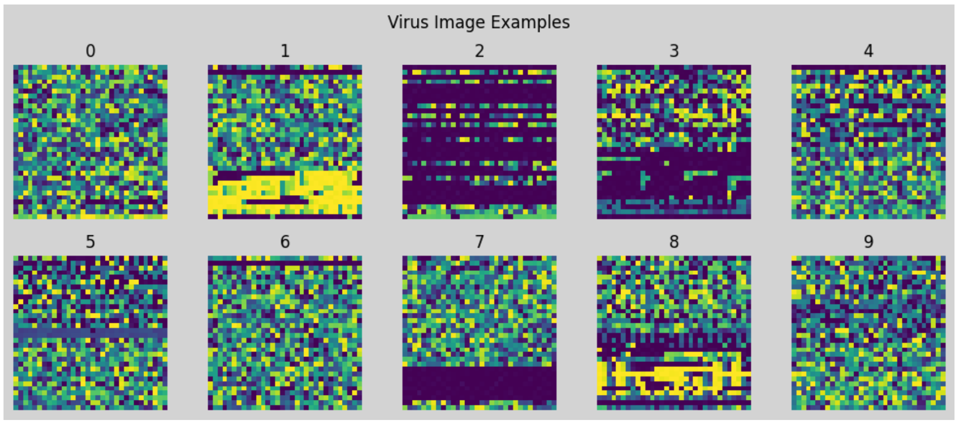


Figure 1: Sample image from each class of the VirusMNIST dataset

Histograms of class labels visualized in Figure 2 reveal a significant imbalance in class distribution. For example, Class 6 includes 14,374 samples, while Class 4 has only 738. This imbalance is present in both the training and test sets and may bias learning algorithms if left unaddressed. To mitigate this, our loss function should be appropriately weighted proportionally to class size while training our models.

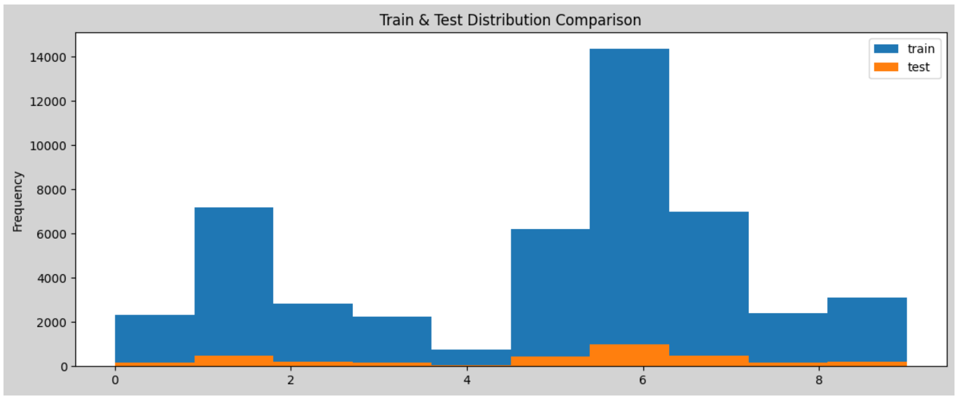


Figure 2: Distribution histogram of the VirusMNIST dataset

2.2 CNN-ViT Model Architecture

Our model architecture, as pictured in Figure 5 in the Appendix, is the same for both the ten-class and nine-class classification problems excluding the output dimension.

The model takes in a 32×32 grayscale image as input. The first part of the network is a convolutional feature extractor made up of three blocks. Each block includes a 2D convolutional layer, followed by batch normalization and a ReLU activation function. This setup helps the model learn non-linear patterns while keeping training stable. The first two convolutional blocks also include max pooling layers, which reduce the spatial dimensions from 32×32 to 16×16 and 16x16 to 8×8. The third block keeps the 8×8 spatial resolution to retain finer local features.

After the convolutional layers, the feature maps are reshaped from (batch_size, channels, height, width) to (batch_size, sequence_length, channels), where the sequence length of 64 corresponds to the flattened 8×8 grid. These features are then linearly projected to a 512-dimensional space. A learnable positional embedding is added to this projection to maintain spatial information for the transformer.

The next component is a PyTorch's `TransformerEncoder` module. It consists of six layers, each with eight attention heads. The input and output dimension of the transformer is set to 512. Each layer includes multi-head self-attention to model long-range dependencies and a feedforward network with hidden dimension 1024. GELU is used as the activation function to improve learning performance.

The transformer outputs are averaged across the sequence dimension using global average pooling. This results in a single vector per sample, containing both local and global features from the original image. Finally, a fully connected layer maps this vector to the final class logits, with either nine or ten outputs depending on the task.

2.3 Transfer Learning

The model trained in the original VirusMNIST paper was done with transfer learning (Noever & Noever, 2021) using the MobileNetV2 model (Sandler et al., 2019). Therefore, we decided to also train a model using transfer learning using a different base model to see if we could improve performance and to determine if a handcrafted model was necessary or efficient for this task. The popular base models we chose were ResNet-18 (He et al., 2015) and EfficientNet-B0 (Tan & Le, 2020).

2.4 Training Algorithm

The dataset used in this work is the same one used in the VirusMNIST paper. Original images, stored as flat vectors, were reshaped to reconstruct their 32x32 pixel, single-channel grayscale image format. The data was subsequently normalized, using a mean of 0.5 and a standard deviation of 0.5 per image. This dataset does not include a validation set, so we reserved 10% of the training set for validation. The model with the best validation accuracy is saved after each epoch.

To train our models, we performed 30 epochs of training with shuffled sampling. Because the samples for each class are very unbalanced, we weighted the `CrossEntropyLoss` function according to the sample count. Training was performed with PyTorch's `AdamW` optimizer with an initial learning rate of 2^{-4} and a weight decay of 1^{-4}. The `CosineAnnealingLR` scheduler was used with a maximum iterations of 30.

Source code for all data analysis and training is available on our GitHub repository.

3 Results

Performance data we gathered from each model includes test accuracy, weighted precision, weighted recall, and weighted F1 score on a reserved test set. The weighted metrics are a weighted mean of the metric over all classes using the support of the class. These results are included in Table 1 for Malware-Only and Table 2 for Malware and Beneware.

Model	Test Accuracy	Precision	Recall	F1 Score
VirusMNIST (Noever & Noever, 2021)	87%	-	-	-
CNN-ViT (Ours)	**90.98%**	**91%**	**91%**	**91%**
ResNet-18-Transfer	89.40%	89%	89%	89%
EfficientNet-B0-Transfer	90.16%	90%	90%	90%

Table 1: Malware-Only Model Performance Comparison

Additionally, the confusion matrix was collected and a ROC curve created for each model and problem combination. These visualizations are included for the top two models, our CNN-ViT for Malware-Only and EfficientNet-B0-Transfer for Malware and Beneware respectively as in Figures 3 and 4. These models are chosen as the best for their respective problem because they achieve the highest scores for all performance metrics. The results for the remaining models are included in the Appendix in Section 5.2. Class 0 is beneware and classes 1-9 are malware as defined in the VirusMNIST paper.

Model	**Test Accuracy**	**Precision**	**Recall**	**F1 Score**
VirusMNIST (Noever & Noever, 2021)	80%	-	-	-
CNN-ViT (Ours)	85.57%	**86%**	86%	86%
ResNet-18-Transfer	85.57%	85%	**86%**	85%
EfficientNet-B0-Transfer	**86.38%**	**86%**	**86%**	**86%**

Table 2: Malware and Beneware Model Performance Comparison

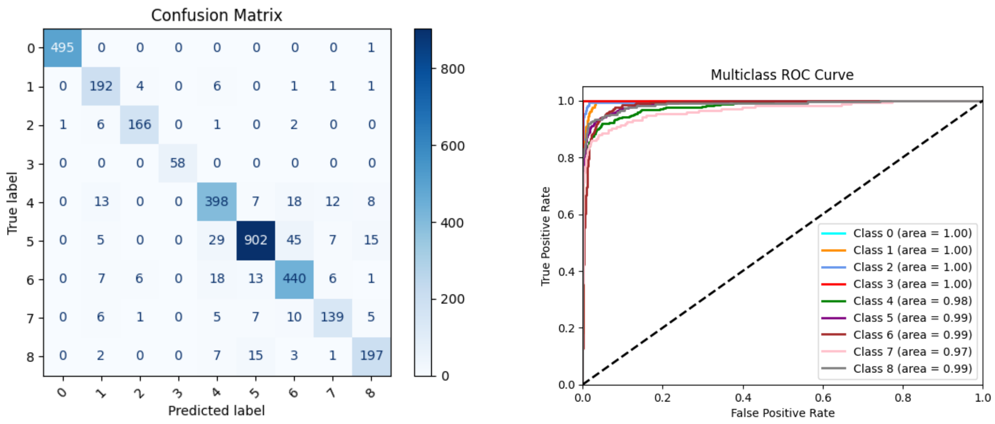


Figure 3: Malware-Only CNN-ViT Results

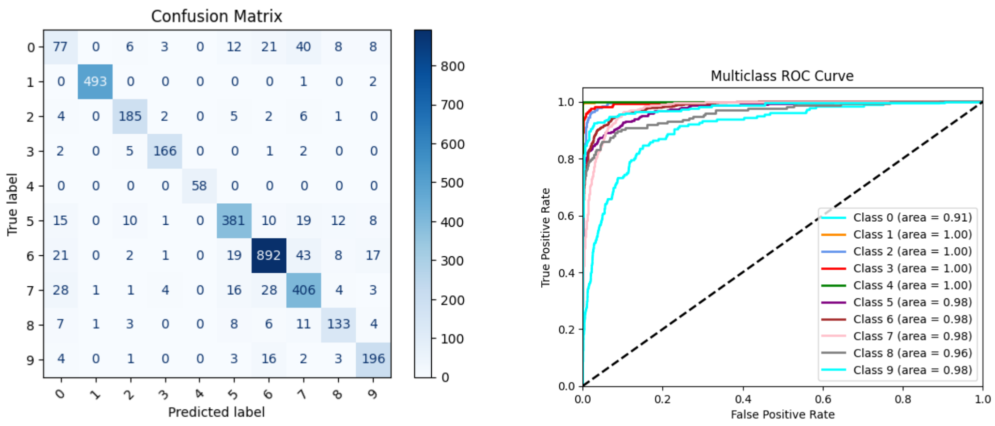


Figure 4: Malware and Beneware Efficientnet-B0 Results

As shown in Figure 4, the most difficult software for the models to classify was beneware; a false positive rate greater than 10% is necessary to achieve a true positive rate greater than 80%, meaning this model on its own is not ready to be at the core of antivirus software, although it may contribute to existing methods. The cause of these results may be because beneware may perform a variety of tasks but each malware family is crafted for a specific task. Future work may include splitting the task into two steps: first classifying malware vs. beneware then classifying families of malware to mitigate this issue.

For the malware-only classification task, models give a much better trade off between the false positive and true positive rates, requiring a much lower false positive rate, i.e. less than 5%, to achieve reasonable malware classification for security researchers.

4 CONCLUSION

In this work, we presented a CNN-ViT hybrid model for detecting malware based on images of executables. With this model, we were able to achieve a higher testing accuracy for classifying malware families than the authors of the VirusMNIST dataset. We also applied transfer learning to this problem using EfficientNet-B0 and were able to achieve higher accuracy in classifying malware families and beneware than the original VirusMNIST authors. Using transfer learning appears to be a better approach to this problem because it achieves great results with minimal developer effort, which is a reasonable trade-off considering our approach beats it slightly for one of the two tasks analyzed. However, as shown in this work, it may not achieve the best performance given that task-specific considerations may be baked into a custom model.

Knowing that these methods outperform those in the VirusMNIST paper, future work includes classifying solely between malware and beneware with the methods described in this work to investigate the efficacy of including this model in an antivirus software.

REFERENCES

Moses Ashawa, Nsikak Owoh, Salaheddin Hosseinzadeh, and Jude Osamor. Enhanced image-based malware classification using transformer-based convolutional neural networks (cnns). *Electronics*, 13(20), 2024. ISSN 2079-9292. URL https://www.mdpi.com/2079-9292/13/20/4081.

Li Chen, Ravi Sahita, Jugal Parikh, and Marc Marino. Stamina: Scalable deep learning approach for malware classification — semantic scholar, Apr 2020. URL https://www.semanticscholar.org/paper/STAMINA:-Scalable-Deep-Learning-Approach-for-Chen-Sahita/54693c4e1933cd9ab535dd87e7358b93b05df496.

Scott Freitas, Rahul Duggal, and Duen Horng Chau. Malnet: A large-scale image database of malicious software, 2022. URL https://arxiv.org/abs/2102.01072.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Pascal Maniriho, Abdun Naser Mahmood, and Mohammad Jabed Morshed Chowdhury. A systematic literature review on windows malware detection: Techniques, research issues, and future directions. *Journal of Systems and Software*, 209:111921, 2024. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2023.111921. URL https://www.sciencedirect.com/science/article/pii/S0164121223003163.

David Noever and Samantha E. Miller Noever. Virus-mnist: A benchmark malware dataset, 2021. URL https://arxiv.org/abs/2103.00602.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. URL https://arxiv.org/abs/1801.04381.

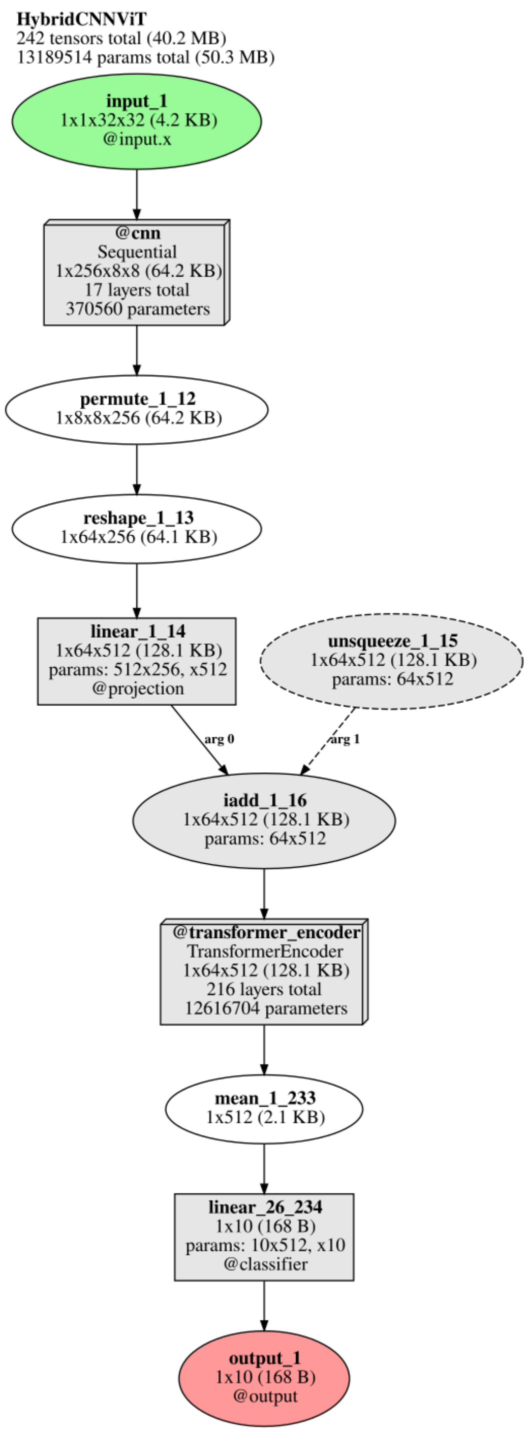
Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. URL https://arxiv.org/abs/1905.11946.

JohnMark Taylor and Nikolaus Kriegeskorte. Extracting and visualizing hidden activations and computational graphs of pytorch models with torchlens. *Scientific Reports*, 13, 2023. ISSN 2045-2322. doi: https://doi.org/10.1038/s41598-023-40807-0. URL https://www.nature.com/articles/s41598-023-40807-0#citeas.

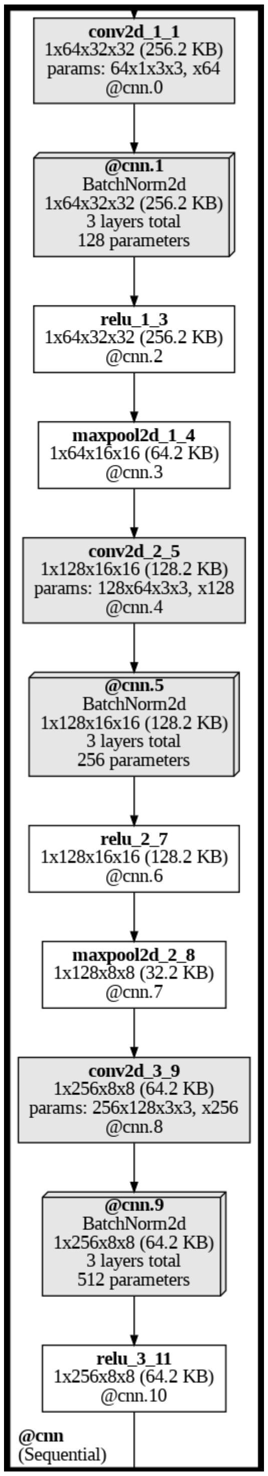
Danish Vasan, Mamoun Alazab, Sobia Wassan, Babak Safaei, and Qin Zheng. Image-based malware classification using ensemble of cnn architectures (imcec). *Computers & Security*, 92:101748, 2020. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2020.101748. URL https://www.sciencedirect.com/science/article/pii/S016740482030033X.

5 Appendix

5.1 Model Diagram



(a) Complete Diagram



(b) CNN Component of Model

Figure 5: CNN-ViT Model Diagram created using `torchlens` (Taylor & Kriegeskorte, 2023)

5.2 Additional Model Testing Results

The remaining testing results for the models that did not perform the best are included as follows:

1. CNN-ViT for Malware and Beneware in Figure 6
2. Resnet-18-Transfer for both problems in Figures 7 and 8
3. EfficientNet-B0-Transfer for Malware-Only in Figure 9

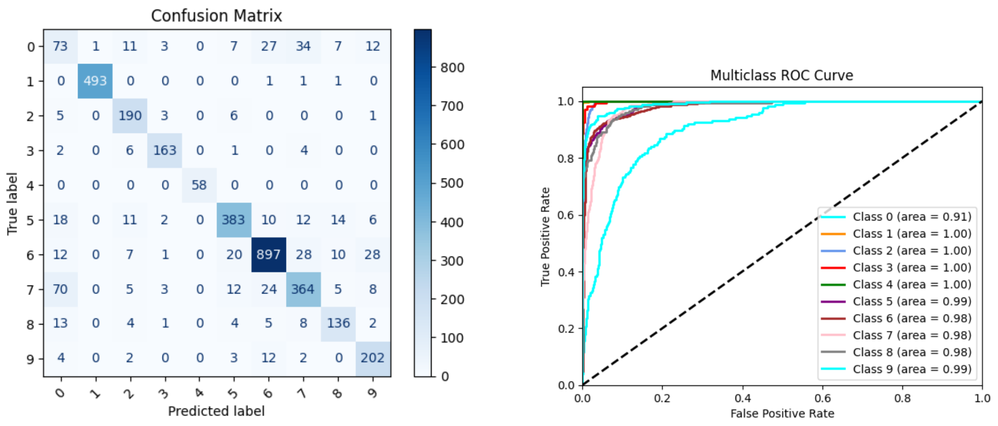


Figure 6: Malware and Beneware CNN-ViT Results

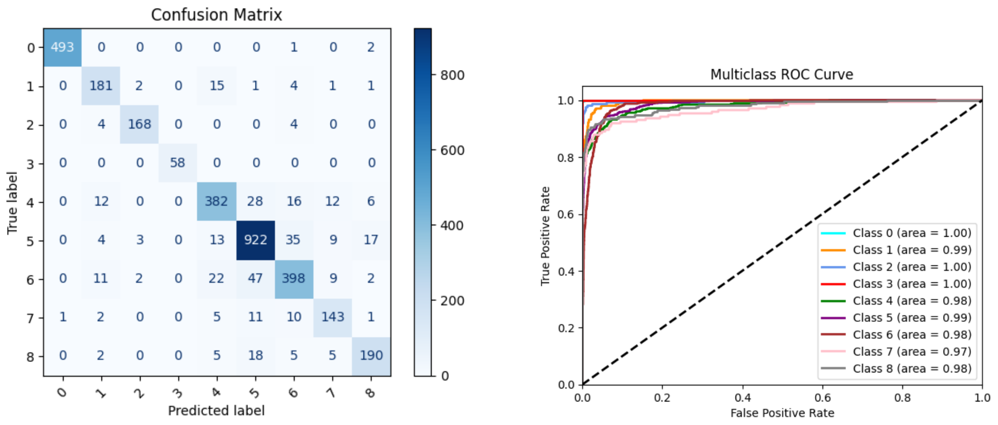


Figure 7: Malware-Only ResNet-18 Results

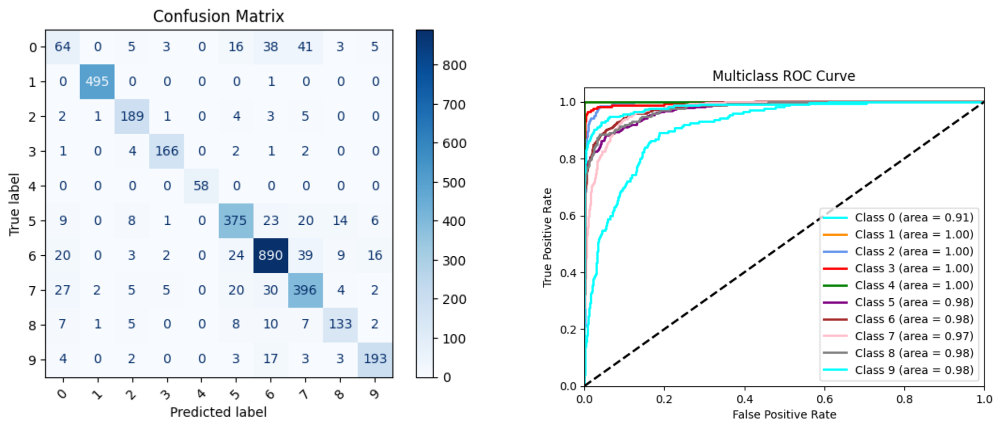


Figure 8: Malware and Beneware ResNet-18 Results

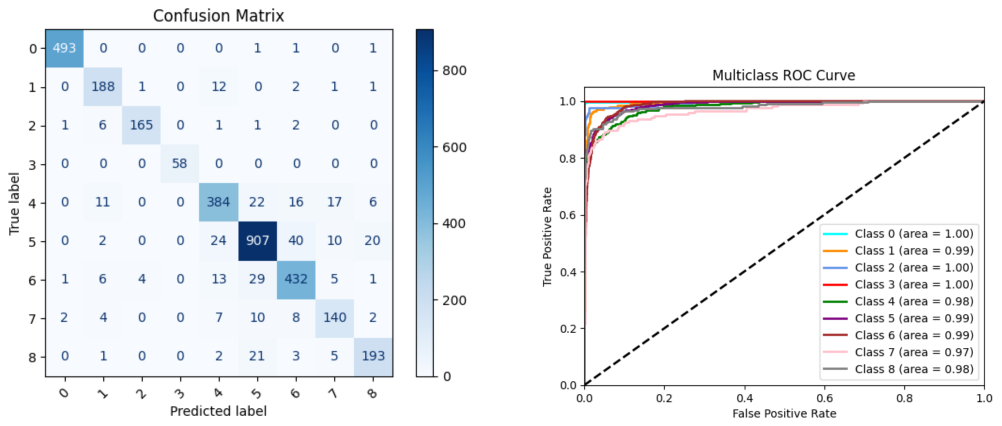


Figure 9: Malware-Only EfficientNet-B0 Results