National College of Ireland

Intelligent Malware Detection Using CNN and OSINT

MSc Research Project
Master of Science In Cyber Security

Anjal Muhammed Madathil Abdul Majeed
Student ID: 23398329

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Anjal Muhammed Madathil Abdul Majeed

Student ID: 23398329

Programme : Master of Science In Cyber Security **Year:** 2025

Module : Practicum

Supervisor : Kamil Mahajan

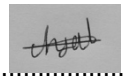
Submission Due Date : 28/01/2026

Project Title : Intelligent Malware Detection Using CNN and OSINT

Word Count : 8265

Page Count : 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: 

Date: 27/01/2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Intelligent Malware Detection Using CNN and OSINT

Anjal Muhammed Madathil Abdul Majeed
23398329

Abstract

High levels of malware sophistication present an ever-growing risk to global cybersecurity. Signature-based detection methods are proving to be less effective for polymorphic/metamorphic threats; therefore, advanced/intelligent techniques are required. This research project evaluates how effective it is to combine visual malware analysis performed through CNNs with contextual data from Open Source Intelligence (OSINT). The primary methodology consists of converting malware binaries into grayscale images these images are then used as visual fingerprints to train the deep learning model. First, a CNN model was created and used for malware family identification. Later, an innovative model that incorporates OSINT prepared the model by combining visual attributes with simulated metadata, such as the threat level of a malware and the propagation mechanism, into a single model. After this was prepared, both models used the Malimg dataset for training and testing, which contained 9,339 instances across 25 families of malware. The results of the experiment showed that the baseline model has high prediction accuracy (test accuracy) of 95.30%. In contrast, by integrating visual information from the analysed image with contextual OSINT information, the proposed OSINT-integrated model achieved improved accuracy (test accuracy) of 95.51% over the baseline model with significant improvements in classifying under-represented and ambiguous families of malware. This research confirms the hypothesis that improving visual analysis with contextual data from OSINT increases the accuracy and reliability of malware detection methodologies, thereby enhancing future directions for building the augmentation OE cyber threat intelligence platforms.

1 Introduction

The modern internet is continually struggling with the onslaught of new threats from the many types of Malware being created and used by Cyber Criminals. From the least harmful forms of Malware such as Adware, to the most destructive types such as Ransomware, to the myriad types of Ransomware created and used by Nation States to commit Cyber Espionage, Malware causes many trillions of dollars in damages every year and uses Compromised Information (Redhu, et al., 2024). Traditionally, the most common way to combat Malware was by using a technique called Signature Based Detection; this technique used a List of Fingerprints of known Malware to identify it. However, as Polymorphic and Metamorphic forms of Malware were developed and deployed, which modified their code to evade Signature-Based Detection, traditional detection techniques became ineffective (Dhanushkodi and Thejas, 2024). With this continuing problem of increasing difficulty, there has been a transition to using Artificial Intelligence (AI) and Machine Learning as the basis for more intelligent and adaptive forms of Detection.

The utilization of artificial intelligence (AI) particularly through deep learning methods is becoming a powerful means of supporting cybersecurity and detecting the more complicated behaviours and patterns of malicious activity (Podder et al., 2021). A new approach for using deep learning to analyse malware is to turn a malware binary into an image. In this method, the malware binary's file structure has been converted into a two-dimensional image so that the latest computer vision models (e.g., convolutional neural networks (CNNs)) can be applied to it. This method will have a considerable advantage because it is able to detect the texture and structure of a particular family of malware without the use of complicated reverse engineering methods or dealing with obfuscated code (Dhande et al., 2024).

Although image-based analysis has its strengths, one downside is that it only uses the characteristics of the binary itself. Thus, there is no information in the analysis to help a cybersecurity analyst deduce the type, source, and impact of the threat. However, such information may be obtained via the use of Open Source Intelligence (OSINT). OSINT refers to the collection and analysis of data from publicly available sources for intelligence purposes, including security blogs, reports on threats, and databases of vulnerabilities (Yadav et al., 2023). OSINT can provide valuable metadata for a particular malware family as it does not only provide the type of malware and its intended targets but also the means by which it propagates and the actors behind the threat (Browne et al., 2024).

The present research addresses the gap between these two domains by studying how these approaches may be integrated into a singular, unified detection model. AI is a well-established technology that utilizes image-based analysis, whereas OSINT is a well-documented technique for compiling and analyzing intelligence related to threats. Despite the disparate origins, the integration of these tools and techniques into a single model remains a developing area. Therefore, the question is: **whether there would be greater accuracy and resilience in the process of malware family classification if OSINT-derived metadata were combined with convolution neural network (CNN) image-based analyses, rather than relying solely on polymorphic characteristics?**

The purpose of this project is to accomplish four primary objectives in order to answer this question:

The first is to create a baseline malware classification model using a standard CNN architecture trained on malware-as-image data; the second is to develop and implement a novel OSINT-aware model integrating both visual features extracted by a CNN and contextual OSINT features into a single architecture. The third goal is to perform a thorough comparative evaluation of the baseline and OSINT-based models using normalized performance metrics. The fourth goal is to analyse the result to ascertain the specific impact of OSINT integration upon a model's classification accuracy, particularly against difficult or unclear malware families.

1.1 Research Contribution

The main contribution of this study is developing and validating an innovative multi modal deep learning architecture that intelligently combines two different yet complementary data sources to improve Malware Detection. The results of this study will provide Experimental evidence to support using contextual intelligence in automated analysis pipelines, ultimately leading to better and more contextually aware Cybersecurity Defenses.

1.2 Report Structure

The structure of the report is as follows: Section 2 contains a review of existing research on malware detection; Deep Learning; and OSINT. Section 3 will explain our Research method, which includes the dataset used, preprocessing methods employed, and evaluation metrics selected. Section 4 will detail the specification for the design of the proposed model

architectures, Section 5 will present the Experimental setup and describe the implementation details. Section 6 includes the results from evaluating and discussing the results. Finally, Section 7 summarises the findings and indicates possible future research directions.

2 Related Work

This section places the current study in context with other studies related to malware detection and identifies other areas of study in cyber security using AI, deep learning, and malware visualisation, and also looks into how OSINT can affect the modern CTI process, which will lead to identifying the gap this study will fill.

2.1 The Development of Techniques for Detecting Malware

The development of malware detection has followed a progression of stages with the traditional approach to malware detection having been done through signature-based detection. This was simply accomplished by comparing the file hash or byte sequences against a list of recorded malware. Signature-based detection is extremely effective to detect already identified malware in terms of time and speed via preemptive identification of malicious software. However, signature-based detection is an approach to malware detection that reacts rather than preemptively alerts on new, unknown or changed malicious software (for example, Redhu et al., 2024). The next approach to malware detection to evolve was the addition of heuristic analysis, where the malicious code is evaluated for indicators of concern or abnormal behaviour. This has subsequently been succeeded by behavioural analysis, where the actions taken by coded programs are monitored in situ at real time. This includes the monitoring of activities on file systems, network connections, and the changes made to the registry. Behavioural analysis is less effective at detecting obscured malware than heuristic analysis, although it is significantly more robust against such malware. Dynamic analysis can also place significant demands on computational resources, and some malware is programmed to actively avoid being executed in sandboxed (isolated) environments and therefore can easily avoid detection using these approaches. These drawbacks in the signature- and heuristic- and dynamic-analysis techniques are driving researchers towards artificial intelligence-based approaches to develop techniques to generalise prior detection methods for unsampled new malware sets through machine-learning methods (for example, Al Siam et al., 2025).

2.2 Artificial Intelligence in Cybersecurity

The field of Artificial Intelligence in Cyber Security is growing quickly and is creating proactive and reactive defenses within the Cyber Security landscape. AI is being integrated into many Cyber Security tasks by applying Machine Learning algorithms such as Traditional Machine Learning (TML) Algorithms (i.e., Support Vector Machine and Random Forest) and Deep Learning Algorithms. Researchers are currently using Deep Learning Models in a wide variety of Cyber Security applications, including Intrusion Detection, Spam Filtering and Malware Classification (Dhanushkodi and Thejas, 2023). Deep Learning has shown great potential because it will allow Researchers/Machine Learning Engineers to automatically learn Hierarchical Feature Representations from Raw Data, thus eliminating the requirement for Engineer Feature Engineering (Dhanushkodi and Thejas, 2023). Additionally, Continual Neural Networks (CNN) have achieved State-of-the-Art results in Engineering Tasks associated with Structured Data (i.e., Images and Text), thus providing a possibility for Innovative Outcomes in relation to Malware (Dhanushkodi and Thejas, 2023).

2.3 Malware Visualization and CNN-based Classification

The concept of Malware Visualization by converting malware executables to images and using classified methods such as Convolutional Neural Networks (CNNs) has recently come to light. This method transforms the byte stream from an executable into a one-dimensional pixel intensity array, which can then be converted into a two-dimensional grayscale image (similar to what a scanner would do). By using this method, different malware families will each have a unique visual texture or Identifier ("fingerprint"). The main advantage of the approach is that it minimizes the effects of common obfuscation techniques, such as packing and encryption, as these techniques will create distinct visual textures that can be recognized by a CNN (Dhande et al. 2024). The organization of an executable file has a predetermined structural layout including different sections, or sections (for example, .text, .data and .rsrc), which will result in the same visual pattern being reflected in the images created from an executable. This means that CNNs are well-suited to analysing images produced from executables, as CNNs can learn how to identify the hierarchical structure of different visual patterns. As evidence of the potential of this approach, a number of studies have demonstrated this methodology's potential by obtaining high classification accuracy on multiple benchmark datasets, including Malimg and BIG 2015.

2.4 Open Source Intelligence for Cyber Threat Intelligence

Cyber Threat Intelligence (CTI) refers to the collection and analysis of data related to cyber threats to make better informed decisions when defending against such threats. OSINT can be seen as an integral part of CTI because it provides many valuable and timely sources of information about a given threat. In general, OSINT is sourced from freely available public resources (Saeed et al., 2023). Examples include, but are not limited to, security vendor publications, academic journals, dark web sites, social media posts and code repositories. OSINT is unique in that it provides many additional contextual aspects around the information of a cyber threat; these aspects include; the most likely infection vector used by a malware, its command-and-control infrastructure, who is behind the malware and what are their intended strategic objectives (Furumoto et al., 2025). The massive amount of unstructured data produced by OSINT comes with a unique set of challenges associated with the automated processing and incorporation of unstructured OSINT into an overall CTI program; as such, the processing and incorporation of OSINT is an active area of research (Al-Yasiri et al., 2025). As more and more organizations begin to rely on OSINT for CTI purposes, especially through the use of AI, Automated Learning, and Natural Language Processing (NLP),(Alturkistani and Chuprat, 2024) structured intelligence can be separated from unstructured text; Therefore NLP is becoming increasingly important in the extraction of structured information from OSINT text.

2.5 The Integration of AI into a New OSINT System

The combination of AI with the capabilities of OSINT (Open Source Intelligence) represents a growing area that seeks to build security systems that have greater intelligence and situational awareness (Evangelista et al., 2021). Most of the research related to this area revolves around using AI as a means of automating the data collection and analysis process for OSINT. Machine learning models can be used to analyze threat feeds, blogs, and other forms of information to detect new indicators of compromise (IoC), or attribute to previously identified threat groups (Bratsas et al., 2024), thus providing for a more adaptable and active CTI ecosystem than currently exists. Within current models, AI-centric analysis (for instance, malware classification) and OSINT related context are largely viewed as two distinct, disparate phases within the analytic workflow, and as such, require analysts to classify

malware via one method, and subsequently search for OSINT related context manually post classification. In contrast, it is only a moderately explored, however extremely valuable, concept to directly combine OSINT related data with the feature set of AI-based classification systems (Browne et al., 2024).

2.6 Summary and Research Gap

This research highlights the potential of convolutional neural networks (CNNs) in recognizing visual patterns associated with malware families. The use of open-source intelligence (OSINT) provides needed context regarding threat intelligence. Although the literature supports the use of OSINT to provide context for threat intelligence; there exists a substantial research gap in the deep integration of these two paradigms. Current research has either involved visual analysis in isolation using CNNs or applying artificial intelligence (AI) techniques to process textual information contained within open-source intelligence (OSINT). The primary contribution of this research is to propose and evaluate a multi-modal CNN model to provide more accurate classifications of malware families through a late fusion approach that utilizes OSINT features and visual representations (i.e., graphical images) to create a more comprehensive understanding of malware families. This research will directly address the research gap by developing and empirically demonstrating an integrated system that utilizes both visual representation and text-based features.

3 Research Methodology

This section describes how science was used to find answers to some questions. It talks about what data was used, how data was manipulated or changed, what was done with the data, and what will be the official way to see if the proposed systems work correctly.

3.1 Describing Data

The investigators used a dataset maintained by Malimg, which is known to be one of the largest and most comprehensive datasets for studying the characteristics of visualized malware images. The dataset contains 9339 samples of malware from 25 different families. The malware samples were created first and then converted to a grayscale 8-bit image format. The number of samples from different families is unbalanced and therefore provides a good, difficult challenge to machine learning. For example, the Allaple .A family has 2949 samples, while only 81 are in the Skintrim.N family. This made it important to take this into account when designing the training and evaluation methods. A pre-established dataset that has already been set aside for training (7459), validation (923), and testing (957) was also used. Using a pre-established dataset categorizes training of the machine learning system on the most significant number of samples, tuning it through separate datasets during validation to reduce overfitting errors, and testing on new unseen samples to show how well it will perform in the real-world.

Table 1: Breakdown of Malware Families and Sample Counts in the Training Set

Family Name	Train Count	Family Name	Train Count
Adialer.C	97	Lolyda.AA3	98
Agent.FYI	93	Lolyda.AT	164
Allaple.A	2359	Malex.gen!J	108
Allaple.L	1272	Obfuscator.AD	113
Alueron.gen!J	146	Rbot!gen	126
Autorun.K	84	Skintrim.N	64
C2LOP.P	116	Swizzor.gen!E	102

C2LOP.gen!g	148	Swizzor.gen!I	105
Dialplatform.B	132	VB.AT	326
Dontovo.A	124	Wintrim.BX	77
Fakerean	306	Yuner.A	640
Instantaccess	344	—	—
Lolyda.AA1	158	—	—
Lolyda.AA2	137	—	—

3.2 Data Preparation and Preprocessing

In order to effectively prepare the image data for use within the deep learning models, a systematic approach was created to ensure that all images were pre-processed in a consistent manner and could be applied to any of the current CNN model architectures that have recently been created.

- Image Loading: All images were processed through the OpenCV imaging library to allow for easy access to the image files. The images were initially in the grayscale format and thus were loaded as single channel images.
- Resizing: All of the images within the Malimg data set contain different sizes. Hence, to allow for each image to have the same size when inputting them into the neural network, all images were resized to be 224 x 224 pixels. 224 x 224 is a common size to provide for pre-trained models (i.e., ResNet) and provides the best compatibility for future use with transfer learning.
- Color Space Conversion: Images were initially stored as monochromatic (grayscale) images, but pretrained CNN architectures such as ResNet50 accept three-dimensional (RGB) color space images. To enable the use of these architectures, the grayscale images were converted to three-dimensional images by duplicating the grayscale image three times.
- Tensor Conversion: The processed images (NumPy array format) needed to be converted to PyTorch Tensors. Tensors are the primary data structure used in PyTorch for performing mathematical operations.
- Normalizing Data: Pixels within the Tensor representation of the images were converted from the range of [0, 1] to the range of [-1, 1]. Each channel was normalized to a mean of 0.5 and a standard deviation of 0.5, using a common mean and standard deviation across all three channels. Normalization enables improved stability during the training phase and provides faster convergence for the model.

A custom class titled MalwareDataset was established within PyTorch to manage the implementation and modification of the entire pipeline without the need to reload the image for each modification to the dataset. The MalwareDataset class also allows data to be processed as needed by the PyTorch model.

3.3 Augmentation of Data

To improve the generalization capabilities of models and to prevent overfitting due to class imbalance, all data augmentation techniques applied only to training data. Data Augmentation artificially enlarges the dataset by modifying existing images. The specific transformations performed were as follows:

- Random Horizontal Flipping (50%): This transformation randomly flipped each image horizontally with a probability of one half.
- Random Rotation: Pictures were rotated randomly by a degree between minus fifteen degrees (-15°) and plus fifteen degrees (+15°).

- Randomly Resized Crop: This transformation sampled at random and then cropped a portion of the image before resizing it back to its original size of 224 pixels by 224 pixels. It helps to encourage the model to learn from all areas of malware and not just from their most apparent characteristics.

Data augmentations assist in ensuring that the model learns patterns associated with a malware family rather than simply remembering how the training examples were oriented or sized.

3.4 Experimental Setup

The experiments performed for this study utilised a python environment using the PyTorch framework for deep learning. Some of the major libraries used for this experiment were torchvision (for performing computer vision), sklearn (for evaluating the performance), pandas (for storing and manipulating data), matplotlib/seaborn (for visualising the data). All training and inference of the models were conducted on a CUDA-enabled NVidia graphics card, enabling quicker processing time for computing. There were two main models built and trained in the experiment, one being a baseline Convolutional Neural Network (CNN) and one being an OSINT fused model. Both models utilised the same three main building blocks: loss function, optimiser, and learning rate scheduler in order to have them compared fairly.

3.5 Evaluation metrics

In order to evaluate models comprehensively with a quantitative metric for modelling performance we used a standard set of classification metrics. These metrics are as follows:

- Accuracy: This represents the number of correctly categorised samples divided by the total number of all sample counted and gives us an overall sense of how well the model classified correctly.
- Precision: This shows how many true positive predictions made for a particular category were correct out of all positive predictions made for that category and indicates that if it has high precision there are fewer false positive predictions being made.
- Recall (Sensitivity): The recall is derived from taking true positive predictions divided by the total number of actual positive samples counted for that category and indicates that if it has a high recall score it is making less false negative predictions than those which it made true positive predictions.
- F1-Score: The F1 score takes the harmonic mean of precision and recall, therefore it allows to combine both measures into one score when comparing models trained on very unbalanced datasets. Thus, model comparisons across unbalanced datasets are easier due to the availability of only one score combining the 2 key components.
- Confusion Matrix: A confusion matrix displays the results of how many of the model's predictions were true or false, and is also useful for identifying classes for which the model has particular difficulty in distinguishing. We calculated metrics from the results of all predictions for the unseen test set, which allowed us to objectively evaluate the final model performance. The weighted average was used for measuring precision, recall, and F1-score; in order to not penalize partially correct responses from all categories.

4 Design Specification

In this section, the research has produced multiple models with various degrees of complexity; ranging from the most basic CNN architecture to the more complete, multi-modal architectures incorporating OSINT data.

4.1 Baseline CNN Architecture

The baseline model is a custom Convolutional Neural Network used as a benchmark for evaluating model performance based solely upon visual features. The architecture is structured as a linear series of layers consisting primarily of convolutional layers and pooling layers that enable the extraction of progressively more sophisticated hierarchical feature representations from the input images.

The architecture has a number of components critical to its implementation:

- Convolutional Blocks - There are four-layered blocks within the overall architecture. Each of these blocks has been built using sequential Convolutional Layers (Conv2d), Batch Normalization Layers (BatchNorm2d), Rectified Linear Units (ReLU) and BN&MAX Pooling layers (MaxPool2d).
- Convolution Block 1: An input image of size 3 x 224 x 224 is passed to Convolution Block 1, which uses 64 filters of size 3 x 3.
- Convolution Block 2: In Convolution Block 2, the output feature maps generated from Convolution Block 1 are passed through 128 filters, as was done in Block 1.
- Convolution Block 3: As with the first two blocks, Convolution Block 3 makes use of 256 filters.
- Convolution Block 4: Continuing the trend of increasing the number of filters applied to subsequent convolution blocks, Convolution Block 4 uses 512 filters.

The number of filters in the architecture increases with each layer to provide the network with a further understanding of complex patterns, and batch normalization is applied following every convolutional layer to aid in stabilizing training and decreasing the number of internal covariate shifts. Max pooling helps with downsampling the feature maps to lower their compute requirements while further allowing for some amount of spatial shift invariance to be captured within the model.

After the final convolutional block (C4), a global average pooling layer (AdaptiveAvgPool2d) is applied. This pooling layer reduces every 512-channel feature map down to a single value, thus allowing for the creation of a 512-dimensional feature vector. This technique allows for the reduction of the number of model parameters and the prevention of overfitting compared to utilizing multiple fully connected layers.

The 512-dimensional feature vector is forwarded to a classifier head. The classifier head contains a dropout layer at 50% for regularization, then a linear layer with ReLU activation which maps to a 256-dimensional feature space, followed by another further dropout layer at 30%, then a final linear output layer with logits corresponding to each of the 25 total malware classes.

The architecture of the Baseline CNN model is shown in Figure 1.

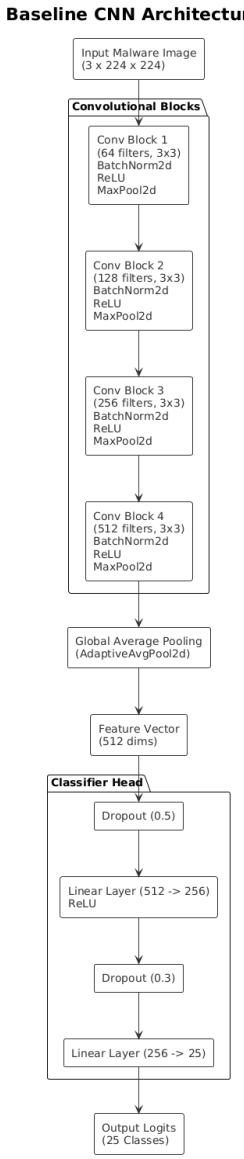


Figure 1: CNN Architecture Diagram

4.2 OSINT Feature Engineering

The incorporation of OSINT information into a model is one of the key features of this advanced version of the model. This project simulated how the characteristics of OSINT could look if they were actually taken from the world of threat intelligence. The benefit of this methodology is that it provides the opportunity to conduct a controlled study to validate the architecture of the data fusion model. For each malware family, a set of characteristics represented in the form of a 10-dimensional feature vector was developed. The 10 defining characteristics of a malware family are comprised of both categorical and numerical information related to the malware families' behavior and history. For example, the following are defined in a 10-dimensional feature vector:

threat_level: A Threat score of 1 to 10.

propagation_method: Categorical (e.g. email, network worm, USB)

encryption_type: Categorical (e.g. none, AES, custom)

target_os: Categorical (e.g. Windows XP, Windows 7, All).

network_activity: Binary (yes or no)
file_manipulation: Binary (yes or no)
registry_changes: Binary (yes or no)
persistence: Binary (yes or no)
privilege_escalation: Binary (yes or no)
first_seen: The numerical year in which the malware was first found.
For each family, these characteristics were randomly generated and stored in a JSON file. When loading the data, the 10-dimensional characteristic vector corresponding to the family is loaded and converted to a PyTorch tensor to serve as input into the OSINT-aware model.

4.3 Architecture of an OSINT-Based Fusion Model:

The multi-modal architecture used with the OSINT model processes two different types of information - a malware image and the associated OSINT feature vector about it. The architecture uses "transfer learning" to process the malware images and it also employs a fusion mechanism to combine OSINT data with the features extracted from the image.
The architecture consists of three main components:
Backbone Image Feature Extraction Component: For this component, the ReseNet50 Model is used as the backbone of the Image feature extraction component. The ResNet50 Model is a powerful Deep Convolutional Neural Network (CNN) that has been extensively trained upon the very large and diverse ImageNet dataset. The pretrained weights of the ResNet50 Model enable the OSINT Model to utilize and leverage rich, low-level features learned from the images in the ImageNet dataset, allowing the OSINT Model to pass the malware image through the Resnet50 Backbone Image Feature Extractor Component. As the Classification Layer (fc) from the ResNet50 Model has been removed and replaced with an Identity layer on the output of the Backbone Image Feature Extraction Component passed through the ResNet50 Backbone Image Feature Extractor Component, when a 224x224 pixel malware image is passed through the ResNet50 Backbone Image Feature Extractor Component, at the output the OSINT Model creates a 2048-dimensional feature vector that describes the high-level visual characteristics of the malware image.
The OSINT framework combines the 10-dimensional OSINT feature vector with the 2048-dimensional image feature vector directly into the Fusion Stage of the model.
The Fusion mechanism is at the heart of this model; it connects both images and OSINT to create a single 2058-dimensional Fusion Feature Vector. The Fusion Feature Vector has both image and OSINT features and will later be combined into a fully-connected Classifier Head that outputs logits for 25 different classes (via a fully connected neural network).
This model's structure is based on a Late-Fusion approach. This allows the model to capture more subtle relationships between visual patterns found in malware images and the additional context of OSINT data.
The Architecture for this model is presented in Figure 2.

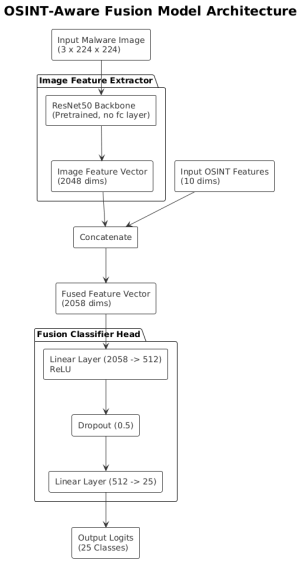


Figure 2: OSINT Architecture Diagram

5 Implementation

The practical application of the Research Design, including the Software Environment, the Data Handling Pipeline, Model Training Methodology and provision of a Web Application ready for Download, will be further explored in this section.

5.1 Development Environment and Libraries

Python 3 has been used for the development of this project. There are three (3) open-source software libraries that provide the basis of this implementation, i.e.,

- PyTorch: The PyTorch software library offers a deep learning framework and a flexible environment through which to define, train and evaluate neural network models.
- Torchvision: This is a library that offers various pre-trained deep learning models (e.g., ResNet50) and offers access to a variety of datasets and common image transformations that can be used with PyTorch.
- OpenCV (cv2): cv2 is an Open Source Computer Vision Library that can be used to perform many of the basic operations of Image Processing e.g. Reading, Resizing, and Colour Space Conversion.

Moreover, the Python Machine Learning Library, aka Scikit-learn, has several machine learning models (both supervised and unsupervised) that can be evaluated using performance metrics such as Accuracy, Precision, Recall, F1 score and Confusion Matrix.

Additionally, NumPy (Numerical Computation) and Pandas (Data Manipulation) libraries are used to perform all of the numerical calculations required for creating Machine Learning Models as well as to perform data re-shaping tasks.

For visualization of the generated model results, Matplotlib and Seaborn provide the tools to generate graphical representations of the results in the form of training curves, heatmaps and so on.

Finally, Flask is an easy-to-use web framework for developing a very basic web application to deploy and interact with the Machine Learning models developed. The computational operations were completed using a system that included an NVIDIA GPU and the CUDA Toolkit was used to boost the performance of the PyTorch operations being carried out on the GPUs.

5.2 Data Loading Pipeline

Custom PyTorch Dataset implementations were used to facilitate the efficient loading of the dataset from Malimg.

MalwareDataset is the Dataset used for the baseline model. All paths to images and the labels were indexed when it was instantiated. The __getitem__ method is used to retrieve the appropriate image from the Dataset by its index and process it (resize, convert to tensor, normalize) as needed, as well as apply augmentation if requested.

MalwareDatasetWithOSINT is a subclass of MalwareDataset which supports the improved OSINT-aware model. In addition to loading and processing the image, this Dataset retrieves the 10-dimensional OSINT feature vector that corresponds to the malicious software family of the image. It returns a triplet for every sample; specifically, it returns the feature tensor from the image, the OSINT feature tensor, and the label for the sample.

These custom datasets were packed into PyTorch DataLoader instances. The DataLoader takes care of batching the dataset, shuffling the training dataset to reduce variance with each epoch, and utilising multiple worker processes to load the data in parallel. By doing so, the DataLoader prevents the data loading process from being a bottleneck during the training phase. A batch size of 32 was used for all experiments conducted.

5.3 Training Strategy

The neural networks were trained using a custom training function, called train_model, that implemented the complete training and validation process for a given number of epochs.

The CrossEntropyLoss function was chosen as the loss function because it is commonly used for multi-class classification problems. In order to handle the major class imbalance present in the Malimg dataset, class weights were calculated and assigned to the loss function. The class weight was based on the inverse of the class frequency observed in the training data, and therefore, this approach applied a heavier penalty to misclassifications of the minority classes and therefore encouraged the model to focus more on them.

The trained models were optimized using the AdamW optimizer, which is a variant of the Adam optimizer designed to improve regularization by providing improved weight decays and provide the potential to generalize better than the Adam optimizer. The AdamW optimizer has a learning rate equal to 0.001 and a weight decay equal to 1e-4.

To ensure an optimal training process, a combination of standard practices that promote nomenclature and universalization as well as preventing overfitting has been utilized for this work. A Cosine Annealing Learning Rate Scheduler was used to dynamically change the Rate of Learning during the training process; the Rate of Learning was reduced over a series of epochs following the shape of the Cosine curve. By employing this strategy, the model will be more likely to reach a global extremal point on the Loss Landscape.

In addition, an Early Stopping mechanism was utilized to reduce the potential for Overfitting and to save computational costs. The accuracy of the training set on the Validation set was tracked after each epoch. If the accuracy did not improve over a certain number of consecutive epochs (5 Epochs of "Patience"), the training was stopped, and the model's weights for the epoch that had shown the best Validation Set accuracy were retained.

5.4 Deployment with Flask

For the trained models to be presented in a real implementation of malware classification, the models were built into a Flask (micro) web application. This implementation provides a convenient interface for end users to classify their malware samples.

The Flask web application is implemented with a back-end that loads the weights of the 2 models (the baseline model and the OSINT-aware model) when it starts. It will have many different APIs defined in the backend via Flask, but the main API is /upload where a file can be uploaded by the user and sent to the back-end. The back-end will receive the file, save it, and process it using the same pre-process functions that the training pipeline used and then use that processed sample to perform inference on it with the model that the user selected to perform the malware classification on.

Inference Process: Inference on the baseline model is a straightforward process; however, for the OSINT-aware model, it is more complicated. This is due to the OSINT data not being available from the OSINT vector for that particular sample that is unknown. Therefore, the application takes a departure from the research approach, and instead, takes a more practical approach to the inference of the model on the image uploaded by the user, which is to perform an inference on that uploaded image against the OSINT vector of every available known malware family. The resulting probability distributions from the inference would be averaged together to obtain a final probability prediction of where the image may belong to, allowing the methodology to assess the OSINT-aware model's performance on an actual unknown malware sample.

The Frontend of the application is comprised of a basic HTML/CSS/JavaScript code base that allows users to upload files and select between using the baseline model or the OSINT-aware model. After receiving a predicted classification for the uploaded file from the API, the Frontend will dynamically display the uploaded image alongside a ranked list containing the top five predicted malware families, including their confidence levels and any relevant OSINT metadata.

The deployment of these models provides an opportunity for users to see how the research models can be utilized as part of an operational analysis platform, thereby closing the gap between research and real-world applications.

6 Evaluation

This section provides a thorough examination and assessment of the results from the experiments carried out, providing an evaluation and comparisons of both the results from the baseline CNN and the results from the OSINT-aware model based upon the metrics that are

defined in the methodology section of this thesis. Additionally, each of the results is discussed thoroughly so that the implications of each result are clear.

6.1 Experiment 1: Baseline CNN Performance

The baseline CNN was trained for a maximum of 20 epochs, with early stopping being utilized as the stopping mechanism. The baseline CNN demonstrated considerable learning ability; the best performance on the validation set occurred at epoch 15, with a validation accuracy of 95.12%. After epoch 15, the validation performance plateaued, leading to early stopping being employed after epoch 20.

Next, the best saved model was evaluated against the unseen test set containing 957 test samples. The summary of the final results is:

- Test Accuracy: 95.30%
- Weighted Precision: 0.9408
- Weighted Recall: 0.9530
- Weighted F1-Score: 0.9432

The performance metrics collected indicate that the overall performance of the baseline CNN is exceptional; the majority of the malware samples were classified correctly by the model. The training and validation curves (Figure 3) demonstrate a normal growth pattern throughout the training process; both training and validation loss declined over time, while the accuracy of the model increased.

A confusion matrix and classification report were generated to better understand how well the model performed in classifying the different families of samples. The classification report showed that the model had a strong performance in many of the families, with precision and recall scores of 1.00 for classes such as "Adialer.C", "Agent.FYI" and "Allaple.L". However, it also highlighted some significant areas where the model performed poorly. Notably, the model failed to classify any of the samples from the "Autorun.K" and "Swizzor.gen!I" families, resulting in a precision, recall and F1-score of 0.00 for both. These classes are considered minority classes in the dataset so the model may have had difficulty classifying these samples due to the limited amount of training examples and the fact that these samples may have been visually similar to other more dominant classes.

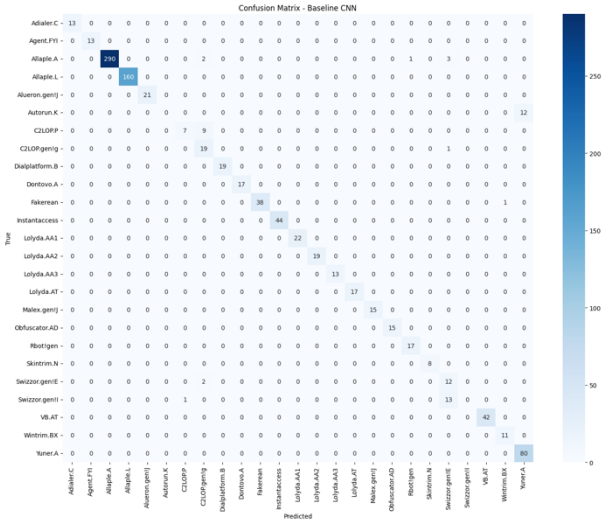


Figure 3: Baseline Model Confusion Matrix

6.2 Experiment 2: Performance of the OSINT-aware Model

The OSINT-aware model was also trained on the same data set using the ResNet50 backbone. The OSINT-aware model began learning much faster than the baseline model during the first few epochs, due to the use of pretrained weights from the ResNet50 backbone. The OSINT-aware model reached its peak validation accuracy of 94.91% during the seventh epoch of training. It was determined that the OSINT-aware model had plateaued in validation performance and training was stopped early during the twelfth epoch.

The final evaluation results for the OSINT-aware model on the test data set are as follows:

- Test Accuracy: 95.51%
- Weighted Precision: 0.9529
- Weighted Recall: 0.9551
- Weighted F1-Score: 0.9515

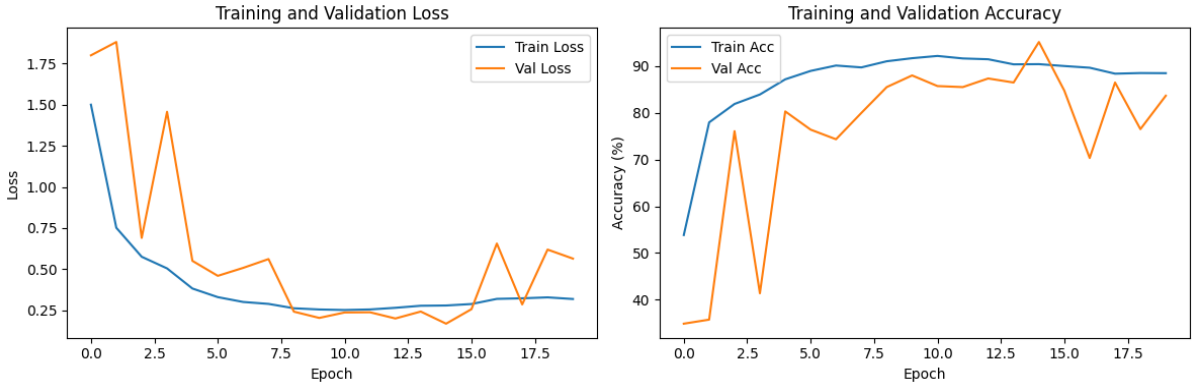


Figure 4: Baseline Model Training and Validation Curves

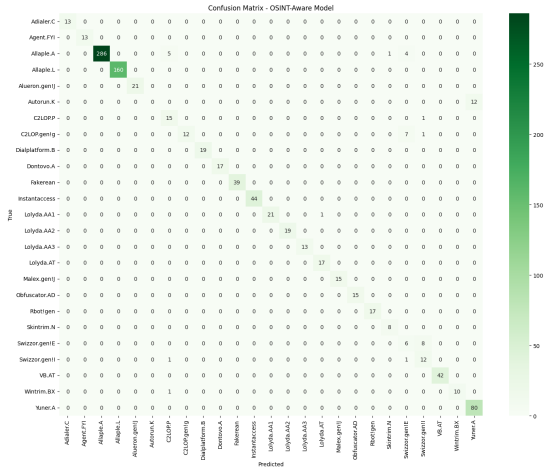


Figure 5: OSINT Confusion Matrix

The OSINT-aware model was able to outperform the baseline in terms of overall accuracy and F1-score. It also exhibited a more tumultuous training and validation curve pattern as seen in Figure 4. However, it was on the basis of the classification report for the OSINT-aware model that provided conclusive evidence of this model being better than the baseline, even though this model also performed poorly on 'Autorun.K' malware family. However, improvements were found by comparing the other classes (that traditionally have low scores on the baseline) when comparing the OSINT-aware model to the baseline model. For example, with respect to the 'Swizzor.gen!I', the F1 score improved from 0.00 to 0.67 while

for 'Swizzor.gen!E', it dropped from 0.56 to 0.38 due to a different precision to recall balance, albeit still an improvement. Most notably, with the OSINT-aware model, the F1 score for 'C2LOP.P' increased from 0.58 to 0.79 due to the inclusion of OSINT additional context, which allowed the model to differentiate between visually similar malware families. The OSINT-aware model could use additional features such as: 'threat_level' or 'propagation_method', to increase the number of models and decrease the number of misclassifications made by the image only model. Additionally, the results indicate that OSINT-aware models have the ability to leverage OSINT features of a given piece of malware to improve accuracy and thus ultimately reduce the number of wrongly classified malware classes. Please see Figure 5 for a graphical representation of the confusion matrix generated by the OSINT-aware model.

6.3 Comparative Analysis

For direct performance of both models, Performance Metrics are shown in Table 2, showing marginally greater performance of the OSINT aware model as compared to the Baseline CNN across all Primary Metrics.

Table 2: Performance Comparison of the Models

Model	Accuracy	Precision	Recall	F1-Score
Baseline CNN	0.9530	0.9408	0.9530	0.9432
OSINT Aware Model	0.9551	0.9529	0.9551	0.9515

Although the 0.21% increase in accuracy appears slight, the real importance of this improvement comes through from the per-class analysis, which reveals that the gains are not evenly spread, but rather, are concentrated in the model's enhanced performance in difficult-to-classify classes. The OSINT data has been shown to have a measurable impact on the model's ability to make better decisions when the visual features were ambiguous. The relative success of the OSINT model reinforces the main hypothesis of the research, that by combining visual and contextual data, the robustness of the classification process is significantly enhanced. The bar charts in Figure 7 further illustrate this comparison graphically.

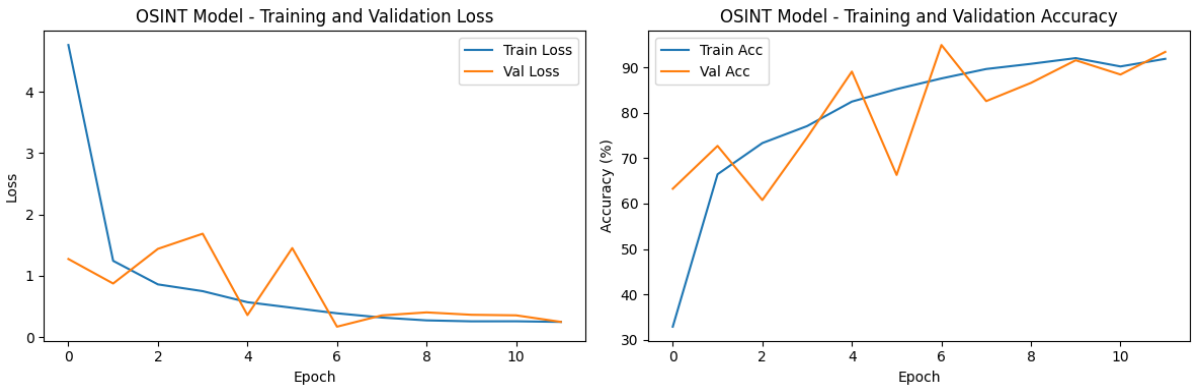


Figure 6: OSINT-Aware Model Training and Validation Curves

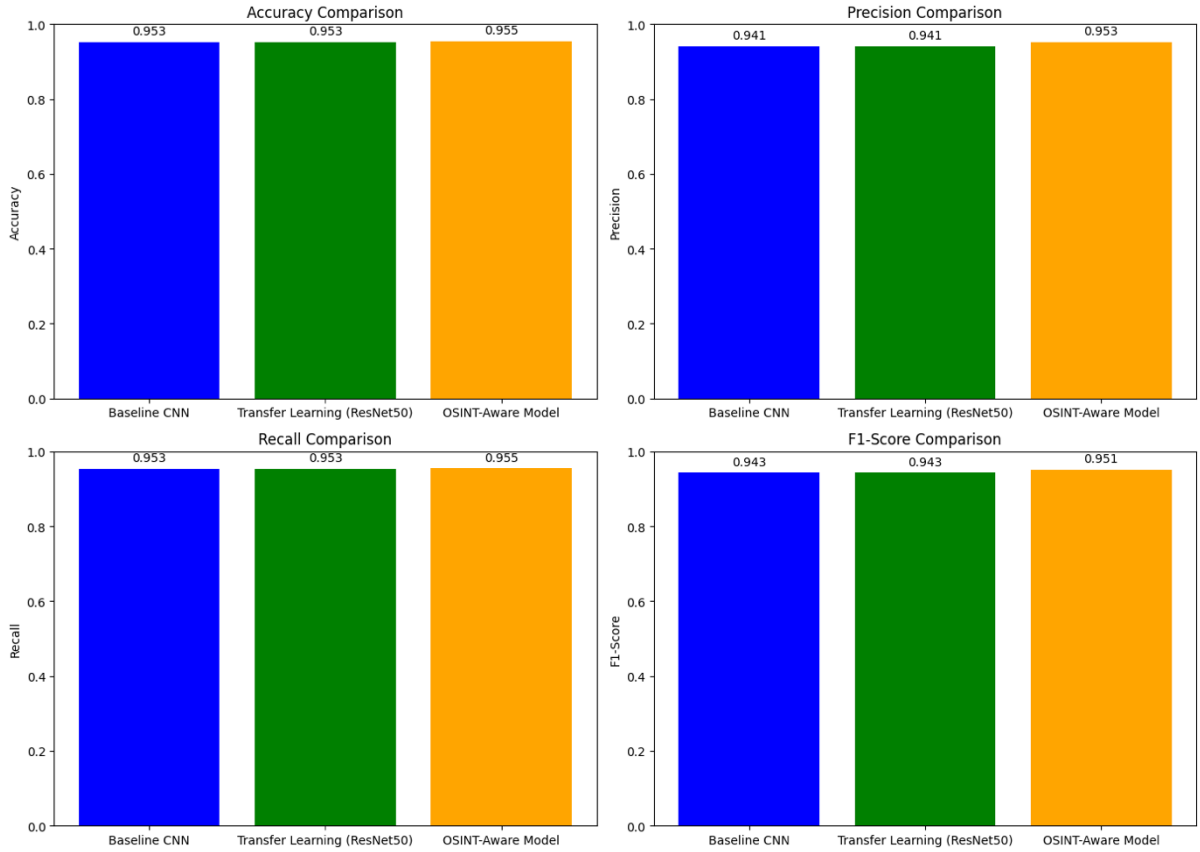


Figure 7: Models Comparision

6.4 Discussion

The results of this evaluation strongly support the overall effectiveness of this OSINT-aware fusion model. The ConvNet baseline produced an almost perfect test result; this is indicative of how powerful image analysis is when working within the Malimg dataset. Obtaining greater than 95% accuracy with a modified ConvNet architecture is an outstanding accomplishment unto itself.

The significant conclusion drawn from this research is that even simulated OSINT data (open source intelligence) results in improved model performance. The OSINT-aware model not only had a slight advantage in total accuracy compared to its predecessor, but also produced significantly higher F1 scores for multiple minority families that were difficult to classify. These results indicate that this model was able to learn to make connections between visual patterns and abstract contextual metadata associated with them. For instance, if two families (A and B) appear identical visually, however family 'A' spreads by email (an OSINT-based metric) while family 'B' spreads using flash drives, the OSINT-aware model will be able to leverage that additional information to yield a stronger prediction. This ability to disambiguate between numerous possible outcomes is a tremendous advantage in practice where many malware developers use recycled code to produce visually similar products across various families.

This study's most important limitation would be the random generation of the simulated OSINT data as the way to create a controlled way to test the architecture and confirm its utility, therefore the gains in terms of performance will be under estimated.

Had the researchers been able to use actual OSINT from threat intel platforms that have created the data, there is no doubt that there would be far greater performance improvements because the correlation between the visual features and the actual metadata would be stronger and more consistent.

The other limitation in this study is that the Malimg dataset was created a long time ago and the visual features of current malware families may be different.

Overall, this study provides clear evidence that the architected concept of using multi-modal models that aggregate intrinsic features of the sample with extrinsic contextual intelligence will allow us to have an effective way of doing automated threat analysis, which is aligned with the current view of CTI that relies heavily on the use of context to interpret and manage threats (Saeed et al., 2023).

7 Conclusion and Future Work

This research project was conducted to determine if combining open-source intelligence with convolutional neural networks and their associated visual analysis could enhance the classification of malware. This goal was achieved by designing, implementing, and thoroughly evaluating two models: (1) A baseline model that used only malware images; and (2) The OSINT-aware fusion model, which utilized both visual and contextual information.

The project was completed and met all objectives. A strong baseline CNN was created with a 95.30% accuracy on the Malimg dataset. A multi-modal OSINT-aware model was created and developed with a ResNet50 backbone to extract image features in combination with a simulated 10-dimensional OSINT feature vector that gave the two modalities a joint existence for analysis. A comparison of the OSINT-aware model against the baseline model demonstrated that the OSINT-aware model outperformed the baseline model at 95.51% accuracy with F1-scores of 0.9515. One of the primary conclusions is that the OSINT data produced the greatest benefit in classifying ambiguous, underrepresented malware families where the baseline model failed, thereby supporting the original research hypothesis. The project presents a validated multi-modal fusion architecture that illustrates the value of adding external, human-centric, domain-specific intelligence to automated analysis.

The project's findings provide significant insight into how the field of cybersecurity will continue to develop in the future. In this regard, current research is providing evidence that rather than developing ever more sophisticated systems that rely solely on a single form of analysis, we should develop systems capable of synthesizing information from a variety of sources (OSINT, Internet data, data mining, etc.). Incorporating the contextual data will provide increased robustness and accuracy when making decisions and lead to developing more trustworthy automated analysis systems.

Despite the encouraging results of this study, there are several avenues to pursue with this research moving forward.

- Real-world OSINT Integration: A significant follow-up on this research would entail replacing the simulated OSINT data with actual threat intelligence that has been curated from legitimate sources. To facilitate that action and expedite future research, a means for automatically collecting and parsing OSINT from aforementioned resources (VirusTotal, security blogs, threat feeds) and linking that data back to the malware samples is needed. The testing of the model's effectiveness under real-world conditions will be of immense benefit.
- Advancing Feature Fusion Techniques: Currently, the model for feature fusion employs basic concatenation as a method of integrating OSINT with visual artifacts.

However, there are far more complex and sophisticated methods of fusing these features together. Proper implementation of these advanced techniques – including the use of attention mechanisms – will allow the model to dynamically weight the visual OSINT attributes depending upon the sample it is classifying.

- Explainable AI (XAI): As the cybersecurity domain becomes increasingly dependent upon Artificial Intelligence (AI), transparency into the decision-making process of these models will be equally as significant. By utilizing some of the more popular XAI methods (SHAP and LIME) on the fused model, transparency into the rationale for its classification will be offered (Zhang et al, 2022) while simultaneously enhancing the trust and utility of this tool set for cyber analysts (Mendes and Rios, 2023).
- The Evaluation of Contemporary Datasets: The testing of the developed architecture on larger-scale datasets of malware known today is required to determine its efficacy at detecting the latest technical trends in malware and obfuscation techniques.
- Zero-Day Threat Recognition: The architecture is presently designed to provide classification of malware. Future efforts could involve modifying the architecture to provide detection of anomalies. This would allow the trained architecture to detect new families of malware that do not fit any established classes, thus combating the problem of zero-day threats.

In conclusion, the research presented in this project provides strong evidence that the blending of contextual and visual data will enhance current methods of malware detection. Furthermore, the OSINT-aware model reviewed in this research offers a unique proof of concept for the creation of a new, more intelligent, and sophisticated cybersecurity defense model.

References

Alturkistani, H. and Chuprat, S., 2024. Artificial intelligence and large language models in advancing cyber threat intelligence: A systematic literature review.

Al Siam, A., Hassan, M.M. and Bhuiyan, T., 2025, February. Artificial Intelligence for Cybersecurity: A State of the Art. In 2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC) (pp. 1-7). IEEE.

Al-Yasiri, J.H., Zolkipli, M.F.B. and Farid, N.F.N.M., 2025. Multilingual Cyber Threat Intelligence Feeds Preprocessing for Threat Intelligence Event Extraction: A Systematic Literature Review. Karbala International Journal of Modern Science, 11(3), p.15.

Bratsas, C., Anastasiadis, E.K., Angelidis, A.K., Ioannidis, L., Kotsakis, R. and Ougiaroglou, S., 2024. Knowledge graphs and semantic Web tools in cyber threat intelligence: A systematic literature review. Journal of Cybersecurity and Privacy, 4(3), pp.518-545.

Browne, T.O., Abedin, M. and Chowdhury, M.J.M., 2024. A systematic review on research utilising artificial intelligence for open source intelligence (OSINT) applications. International Journal of Information Security, 23(4), pp.2911-2938.

Dhanushkodi, K. and Thejas, S., 2024. Ai enabled threat detection: Leveraging artificial intelligence for advanced security and cyber threat mitigation. IEEE access, 12, pp.173127-173136.

Dhande, Y.H., Zade, A. and Patil, S.P., 2024, September. An Empirical Review of Dark Web Data Classification Methods Using NLP, SVM, CNN, and GAN. In 2024 4th International Conference on Computer, Communication, Control & Information Technology (C3IT) (pp. 1-8). IEEE.

Evangelista, J.R.G., Sassi, R.J., Romero, M. and Napolitano, D., 2021. Systematic literature review to investigate the application of open source intelligence (OSINT) with artificial intelligence. Journal of Applied Security Research, 16(3), pp.345-369.
Furumoto, K., Morikawa, T., Kolehmainen, A., Silverajan, B., Takahashi, T. and Inoue, D., 2025. A Comprehensive Survey of Threat Intelligence Research: A Measurement-Based Study. ACM Computing Surveys.
Garg, P., Shrivas, N., Kalia, A., Roy, R., Sharma, S. and Agarwal, G., 2025. OSINT: A Double-edged Sword.
Kühn, P., Wittorf, K. and Reuter, C., 2024. Navigating the shadows: Manual and semi-automated evaluation of the dark web for cyber threat intelligence. IEEE Access.
Mendes, C. and Rios, T.N., 2023. Explainable artificial intelligence and cybersecurity: A systematic literature review. arXiv preprint arXiv:2303.01259.
Ofusori, L., Bokaba, T. and Mhlongo, S., 2025. Explainability and interpretability of artificial intelligence use in cybersecurity. Discover Computing, 28(1), pp.1-23.
Podder, P., Bharati, S., Mondal, M., Paul, P.K. and Kose, U., 2021. Artificial neural network for cybersecurity: A comprehensive review. arXiv preprint arXiv:2107.01185.
Redhu, A., Choudhary, P., Srinivasan, K. and Das, T.K., 2024. Deep learning-powered malware detection in cyberspace: a contemporary review. Frontiers in physics, 12, p.1349463.
Saeed, S., Suayyid, S.A., Al-Ghamdi, M.S., Al-Muhaisen, H. and Almuhaideb, A.M., 2023. A systematic literature review on cyber threat intelligence for organizational cybersecurity resilience. Sensors, 23(16), p.7273.
Santos, P., Abreu, R., Reis, M.J., Serôdio, C. and Branco, F., 2025. A systematic review of Cyber Threat Intelligence: The effectiveness of technologies, strategies, and collaborations in Combating modern threats. Sensors, 25(14), p.4272.
Yadav, A., Kumar, A. and Singh, V., 2023. Open-source intelligence: a comprehensive review of the current state, applications and future perspectives in cyber security. Artificial Intelligence Review, 56(11), pp.12407-12438.
Zhang, Z., Al Hamadi, H., Damiani, E., Yeun, C.Y. and Taher, F., 2022. Explainable artificial intelligence applications in cyber security: State-of-the-art in research. IEEe Access, 10, pp.93104-93139.