Why Does Aggressive Resizing Preserve Malware Image Classification Performance?

Evaluating the Impact of Interpolation and Spatial Detail on Family-Discriminative Signals

Cristina Mitu

Supervisors: Tom Viering, Akash Amalan

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Cristina Mitu
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Akash Amalan, Georgios Smaragdakis

Abstract

Malware binaries can be represented as grayscale images by placing byte values on a two-dimensional grid. Convolutional neural networks can classify such malware images with high accuracy, but it is less clear why this performance can remain strong when the images are aggressively resized. This paper studies this phenomenon by examining how different interpolation methods affect accuracy, whether information learned by a high-resolution model remains useful after aggressive resizing, and which retained pixel values are most predictive for distinguishing malware families. The results show that all studied interpolation methods remain above 0.99 through 8×8. At lower resolutions, nearest-neighbour performs best at 4×4 and 2×2, whereas bilinear and bicubic unexpectedly perform better at 1×1. Blurring reduces accuracy, while a model trained at 224×224 does not transfer reliably when test images are downsampled and then restored to the original input size, indicating that broad texture alone is not sufficient for classification and that successful low-resolution models adapt to the resized representation. Decision-tree analysis further reveals that byte values sampled from fixed relative locations can contain family-specific signal, and backmapping shows that influential sampled locations frequently overlap with parsed binary sections, especially executable code. Still, these findings do not demonstrate semantic code understanding.

The results suggest that aggressive resizing preserves malware-family information through coarse byteplot layout, sampled byte values, and contrast patterns, rather than exact semantic binary regions.

1 Introduction

Malware remains a persistent and evolving cybersecurity threat. For Windows alone, the AV-TEST AV-ATLAS database grew from 920 million registered malware samples in 2024 to 995 million in 2025, showing the continued growth of the malware ecosystem (Selinger, 2026). In this context, identifying the family of a suspicious binary can help analysts understand its likely behaviour, relate it to known threats, and support mitigation or investigation (Aslan & Samet, 2020). However, manual analysis is time-consuming, while automated approaches often rely on handcrafted features such as imports, opcode sequences, entropy, or executable-section metadata. Although effective, these features can be affected by obfuscation, packing, and adversarial modification (Sihwail et al., 2018).

Machine learning can automate parts of this process by learning predictive patterns directly from malware samples. However, high accuracy alone is not sufficient for a trustworthy cybersecurity system. A model may capture meaningful family-level characteristics, but it may also rely on shortcuts such as file size, padding, packing artefacts, or biases in the dataset. Such shortcuts can make predictions fragile when samples or attacker behaviour change. Understanding *what* a classifier uses is therefore important alongside measuring *how accurately* it performs.

A different approach is malware-image classification, where each byte of a binary file is interpreted as a grayscale intensity value and arranged into a two-dimensional image, commonly called a byteplot (Nataraj et al., 2011). Related malware samples may produce visually distinctive byteplots because they share code regions, resources, padding behaviour, packing artefacts, or broad file-layout patterns. This motivates the use of convolutional neural networks (CNNs), which have achieved strong malware-family classification performance on byteplot datasets (Kiger et al., 2022; Vasan et al., 2020).

A practical challenge is that malware binaries produce byteplots with different native dimensions, whereas CNNs usually require fixed-size inputs. Byteplots are therefore commonly resized before classification, often to dimensions such as 224×224 (Kiger et al., 2022; Kumar & Janet, 2022). Resizing is usually treated as preprocessing, but it can substantially change the available information. At very low resolutions, each pixel summarizes a large region of the binary, while interpolation may either preserve selected byte-derived intensities or average neighbouring values into smoother patterns. This raises an important interpretability question. If a classifier remains accurate after a byteplot is reduced to only a few pixels, it is unlikely to depend on detailed byte-level structure alone. Instead, predictions may be supported by broad layout patterns, byte distributions, padding blocks, file-size-related structure, or other coarse regularities. These signals may be meaningful family characteristics, but they may also be dataset-specific artefacts rather than robust properties of malicious code.

This paper investigates aggressive resizing as a controlled approach to study what information survives in malware byteplots. It compares interpolation methods, evaluates performance across progressively smaller resolutions, examines family-level robustness, tests cross-resolution transfer, and probes the information retained by nearest-neighbour-resized byteplots. The results show that high performance can persist at unexpectedly small resolutions. However, it depends strongly on the resizing method and does not imply that high- and low-resolution models rely on the same representation.

The central research question is:

How do malware-image classifiers maintain accuracy when byteplots are aggressively resized?

To answer this question, the paper addresses the following sub-questions:

1. How do extreme downsampling and the choice of interpolation algorithm affect classification accuracy?
2. Which malware families remain robust under resizing?
3. Can a classifier trained on high-resolution byteplots successfully generalize to low-resolution images?
4. What do nearest-neighbour pixels reveal about the family-discriminative information retained at ultra-low resolutions?

2 Related Work

This section places the present study within the broader literature on malware-image classification and low-resolution visual recognition. It reviews prior work showing that malware binaries can be represented as images and classified effectively with CNN-based models, then examines how resizing is used in existing malware-image pipelines and why this preprocessing step deserves closer attention. It also considers related findings from very low-resolution recognition in computer vision, together with the limitations of standard explainability methods at ultra-low resolutions. Taken together, these strands of work help clarify the research gap addressed in this paper and motivate the experimental questions studied here.

2.1 CNN-Based Malware Image Classification.

Static malware analysis classifies binaries without executing them, often using features such as opcodes, imports, entropy, strings, and section metadata (Aslan & Samet, 2020). These features can support scalable classification, but they also depend on assumptions about which binary properties are informative. Nataraj et al. (2011) introduced grayscale byteplots as an alternative representation by mapping raw byte sequences into two-dimensional intensity grids. Their results showed that malware families can form visually distinctive textures, suggesting that byteplots may capture family-level regularities such as shared code, resources, padding, packing artefacts, or section layout. Later work extended this idea using memory dumps (Dai et al., 2018), texture features (Verma et al., 2020), and multi-channel representations (Fu et al., 2018).

As deep learning matured, CNNs became a natural tool for learning features from malware images. Prior work has used CNN architectures, transfer learning, DenseNet-style models, and ResNet-based classifiers to obtain strong malware-family classification performance (Cui et al., 2018; Kumar & Janet, 2022; Vasan et al., 2020; C. Wang et al., 2021). These results show that byteplots contain useful predictive signal. However, high accuracy does not by itself explain what signal the model uses. Because byteplots do not have the object semantics of natural images, classifiers may rely on robust family-level structure, but they may also exploit dataset-specific artefacts such as file size, padding, compiler patterns, packing conventions, or collection bias (Gibert et al., 2020).

2.2 Resizing in Malware-Image Pipelines.

Resizing is a standard step in malware-image classification pipelines, as raw binaries produce byteplots with different native dimensions, while CNNs usually require fixed-size tensor inputs. Prior work therefore often resizes byteplots to standard dimensions such as 224×224 or 256×256 before training (Freitas et al., 2022; Kumar & Janet, 2022). However, most prior work treats resizing as a practical preprocessing step and does not justify the choice of interpolation method. Among the studies reviewed, Zhao et al. (2023) are among the few to directly compare interpolation methods in a malware-image classification setting. They report bilinear interpolation as the strongest option in their multi-channel transfer-learning pipeline. However, their work does not investigate why bilinear performs best or how interpolation changes the byte-derived information available to the classifier.

The present study addresses this gap by treating resizing itself as the experimental variable. It systematically reduces single-channel byteplots from 224×224 to 1×1 and evaluates how the effect of nearest-neighbour, bilinear, and bicubic interpolation changes as progressively less spatial information remains. Rather than only identifying the highest-performing preprocessing method, it examines which byteplot signals survive aggressive compression and how these signals support classification.

2.3 Low-Resolution Recognition.

Very low-resolution recognition has been studied in natural-image computer vision. Z. Wang et al. (2016) show that deep networks can classify images smaller than 16×16, while also finding that high- and low-resolution models may learn different features. This suggests that small images can retain predictive signal, but it does not explain what that signal represents.

It is important to note that malware byteplots differ substantially from natural images. A low-resolution natural image may still preserve object shape or scene layout, whereas a heavily resized byteplot is more likely to preserve coarse file-level regularities such as byte distributions, padding blocks, layout patterns, or size-related structure. The present work builds on the distinction identified by Wang et al. by testing whether a model trained on high-resolution byteplots can reuse its learned representation after aggressive downsampling, and by comparing this with models trained directly at each low resolution.

2.4 Explainability at Ultra-Low Resolutions.

Explainability methods such as saliency maps, gradient heatmaps, LIME, and SHAP can help identify which input regions influence a model prediction (Lundberg & Lee, 2017; Ribeiro et al., 2016). These methods are useful for standard-sized inputs, but they become less informative when the input is extremely small. At 4×4, there are only sixteen spatial positions, so a heatmap cannot provide much localization detail. A simple decision tree is therefore useful as a diagnostic probe at tiny resolutions (Breiman et al., 1984). For nearest-neighbour-resized byteplots, each tree feature corresponds directly to one final sampled pixel. This does not explain the CNN itself, but it tests whether the pixels that survive resizing contain family-discriminative information.

2.5 Research Gap and Contribution.

This work fills the gaps left by prior research by treating resizing as the object of study rather than as a preprocessing step. It asks what information survives when byteplots are reduced to extremely small resolutions, how interpolation changes that information, and whether the remaining signal reflects robust family-level structure or dataset-specific regularities. The experiments address these questions through interpolation comparisons, smoothing controls, cross-resolution transfer, and a decision-tree probe of nearest-neighbour pixels.

3 Methodology

This section describes the experimental strategy used to investigate why malware-image classifiers can remain accurate after aggressive resizing. The main analysis aims to investigate which information remains predictive after resizing by comparing accuracy across different resolutions and interpolation methods. Follow-up experiments examine the role of local contrast, differences between malware families, transfer across resolutions, and the information retained by the few pixels that remain after nearest-neighbour resizing.

3.1 Resizing and interpolation analysis

The central experiment evaluates how CNN classification changes as byteplots are reduced to progressively smaller resolutions. The key comparison is between nearest-neighbour, bilinear, and bicubic interpolation. Nearest-neighbour assigns each output pixel the value of one selected source pixel, so it preserves discrete byte-derived intensity values without local averaging. Bilinear interpolation estimates each output pixel from the four nearest source pixels, producing smoother transitions by averaging nearby values. Bicubic interpolation uses a larger neighbourhood, typically sixteen surrounding source pixels, together with cubic weighting, which produces an even smoother estimate and can preserve gradual broad texture more effectively. Consequently, nearest-neighbour retains sharper local contrast, whereas bilinear and bicubic trade some local intensity differences for smoother approximations of the surrounding byteplot structure. A visual illustration of how the three interpolation methods transform representative byteplots is provided in Appendix A.

This comparison distinguishes two possible explanations for performance loss at extreme resolutions. If the main limitation is simply the decreasing number of output pixels, all interpolation methods should deteriorate similarly as resolution decreases. In contrast, if smoothing away local byte-intensity differences is important, bilinear and bicubic interpolation should decline earlier than nearest-neighbour. Nearest-neighbour would then remain stronger because it preserves sharp sampled byte values and local contrast.

3.2 Contrast-loss control

The interpolation comparison alone cannot fully separate the effect of fewer pixels from the effect of smoothing. Therefore, a Gaussian-blur control is used to test whether reducing local byte-intensity contrast damages classification even when broad byteplot layout remains visible.

Gaussian blur replaces each pixel with a weighted average of values in its surrounding neighbourhood, with closer pixels receiving greater weight than more distant pixels. Increasing the blur radius expands this neighbourhood, progressively suppressing sharp local intensity transitions while preserving broader visual structure. Gaussian blur is not identical to bilinear or bicubic downsampling, but all three operations smooth local transitions. If blur reduces accuracy, this supports the interpretation that sharp contrast patterns contribute useful signal beyond broad visual layout. The blur experiment therefore provides supporting evidence for the proposed explanation of interpolation differences.

3.3 Family-level robustness

Aggregate accuracy may hide important differences between malware families. The family-level analysis therefore measures how classification accuracy changes with resolution for each class separately.

This experiment tests whether low-resolution robustness is shared across families or concentrated in a subset of classes. Families that remain accurate at very small resolutions may contain stronger or more distinct coarse layout, padding, byte-distribution, or resource-related patterns. Families that decline earlier may depend more on local or less distinctive information.

3.4 Cross-resolution transfer

The cross-resolution experiment tests whether a model trained on original 224×224 byteplots can use the same evidence after fine spatial information has been removed. Test images are first downsampled to smaller target resolutions and then restored to 224×224 before inference.

The experiment is repeated with nearest-neighbour, bilinear, and bicubic interpolation. The same interpolation method is used for both downsampling and upsampling, so every input presented to the model has the same final dimensions as the training images. This isolates the effect of information loss and reconstruction style from changes in input size.

If the high-resolution model remains accurate, the information it uses is sufficiently broad to survive resizing. If performance declines while models trained directly at low resolution remain accurate, successful low-resolution classification requires adaptation to the resized representation rather than direct reuse of high-resolution features.

3.5 Nearest-pixel interpretability

A decision-tree analysis is performed at 4×4, 2×2, and 1×1 to examine how much family-discriminative information remains when nearest-neighbour resizing retains only sixteen, four, or one sampled pixel values. Each final pixel value is treated as a separate input feature for a decision tree. The aim is not to reproduce the CNN decision process, but to test whether the small set of retained values alone can still support meaningful family classification and to identify which sampled positions are most informative.

For the 4×4 representation, pixel importance is calculated using the decision tree's normalized mean decrease in Gini impurity. A pixel receives higher importance when splits using that pixel produce a comparatively large weighted reduction in class impurity, particularly when those splits affect many samples or occur near the top of the tree. The importance values are normalized to sum to one across the sixteen pixel features. They therefore describe the fitted decision tree, rather than CNN attention, causal byte importance, or the contribution of a pixel to a fixed proportion of predictions.

The 4×4 representation is analysed in more detail because it retains enough positions to examine their relative contribution while remaining highly compressed. The most important tree pixels are mapped back to their source coordinates in the original byteplot and corresponding byte offsets in the raw binary. The mapped offsets are then compared with parsed

binary sections to examine whether influential pixels repeatedly fall within consistent file regions, such as executable code, data, import-related structures, or other identifiable sections. This analysis helps relate the family-discriminative signal captured by the decision tree to broad binary regions, while not assuming that one fixed section is uniquely responsible for any individual family.

4 Experimental Setup

This section describes how the experiments were implemented in practice. It outlines the software libraries used, how the main interpolation study differs from the follow-up analyses, and how the resized images, CNN models, decision trees, and backmapping procedure were handled consistently throughout the research.

4.1 Dataset

The experiments use grayscale byteplot images derived from malware and benign binaries. Each byte is represented as an intensity value between 0 and 255, and the byte sequence is reshaped into a two-dimensional image before resizing.

The full manifest contains 20,020 image entries, including unpacked samples and packed variants. The main analysis uses only the unpacked subset to focus on resizing rather than the separate effect of packing. This subset contains 10,010 images from 14 balanced families, with 715 images per family. It includes 5,720 PE files and 4,290 ELF files, including the benign classes `BenignPE` and `BenignELF`.

4.2 Data splits and evaluation protocol

All splits are performed at the SHA-256 level to prevent the same binary from appearing in both training and test data. This avoids overestimating generalisation through duplicate or near-duplicate image rows derived from the same binary.

The main interpolation comparison uses three-fold grouped stratified cross-validation. Folds are created over unique SHA-256 hashes and then expanded to their corresponding image rows, so each binary is evaluated once by a model that was not trained on that binary. Mean performance across the three folds is reported.

The interpolation, blur, cross-resolution, and family-level analyses use three-fold grouped stratified cross-validation, with mean performance and standard deviation reported across folds where applicable. The decision-tree analysis uses one fixed split containing 7,004 training images, 1,505 validation images, and 1,501 test images.

4.3 Model training and preprocessing

A separate ResNet-18 classifier is trained from scratch for every resolution, interpolation method, and fold. No ImageNet pretraining is used, and the final classification layer is replaced with a 14-class output layer.

Models are trained with cross-entropy loss and Adam using a learning rate of 10^{-4}, batch size 64, and a maximum of 10 epochs. Early stopping monitors validation loss with patience 2 and a minimum improvement of 10^{-4}. The checkpoint with the lowest validation loss is selected for final testing.

The primary interpolation comparison evaluates 13 resolutions:

$$224,\ 160,\ 112,\ 80,\ 64,\ 48,\ 32,\ 24,\ 16,\ 8,\ 4,\ 2,\ 1.$$

Each resolution is evaluated with nearest-neighbour, bilinear, and bicubic interpolation, resulting in 117 independent training runs across three folds. Images are resized explicitly with the selected interpolation method, converted to tensors, and normalized using the ImageNet mean and standard deviation from the torchvision ResNet-18 preprocessing pipeline (PyTorch Contributors, n.d.). The same normalization is used for all conditions to keep the comparison controlled. No data augmentation is applied.

For the blur control, Gaussian blur radii of 0.5, 1.0, 2.0, and 4.0 pixels are applied to the available resolution-specific CNN models.

4.4 Cross-resolution evaluation

For cross-resolution transfer, the models trained on 224×224 inputs in each fold are evaluated on test images whose information is first reduced to each smaller target resolution and then resized back to 224×224. The same interpolation method is used for both the downsampling and upsampling steps. Thus, all images presented to ResNet-18 have the same dimensions as the training data, while retaining only the information available at the target resolution.

The resized images use the same ImageNet normalization as the training data. This setup keeps the model input dimensions and internal feature-map geometry fixed, allowing the experiment to test whether features learned at high resolution remain useful after fine spatial detail has been removed.

4.5 Decision-tree analysis and backmapping

To analyse the information retained by nearest-neighbour resizing, decision trees are trained on the final pixel values of the 4×4, 2×2, and 1×1 representations. Each retained grayscale pixel is used as one input feature. The trees use a maximum depth of 8, require at least 20 samples per leaf, and require at least 40 samples before an internal node can be split. Balanced class weights are used to reduce the influence of small class-frequency differences.

For the 4×4 representation, the most important tree pixels are traced back to the original byteplot locations from which they were sampled. To do this, two auxiliary images are created: one stores the horizontal coordinate of every original pixel, and the other stores the vertical coordinate. These coordinate images are resized to 4×4 using the same nearest-neighbour operation as the byteplot. For each final output position, the resulting coordinate values identify the original (x, y) pixel location selected during resizing.

The corresponding byte offset is computed as

$$\text{byte offset} = y \cdot \text{image width} + x,$$

where *image width* is the native byteplot width of the individual file. For each mapped position, the relative file offset and, where available, the corresponding PE or ELF section are recorded.

To examine whether influential pixels consistently overlap meaningful binary regions, the mappings of the five highest-importance 4×4 tree pixels are aggregated across the analysed samples. For each pixel, the fraction of mappings falling within executable code, data, imports/linkage, other parsed sections, overlays, or section gaps is calculated. This analysis tests whether the sampled values that help distinguish malware families tend to originate from consistent broad binary regions, while not assuming that any individual pixel represents a unique semantic component.

5 Results

This section presents the results of the experiments described in the methodology section. The interpolation, blur, cross-resolution, and family-level analyses report mean accuracy across three folds. The nearest-pixel decision-tree analysis uses one fixed split.

5.1 Resizing and interpolation analysis

Figure 1 shows that **malware-family classification remains highly accurate after substantial resizing**. All three interpolation methods remain above 0.99 accuracy through 8×8, indicating that the dataset retains strong family-discriminative information even after most spatial detail has been removed.

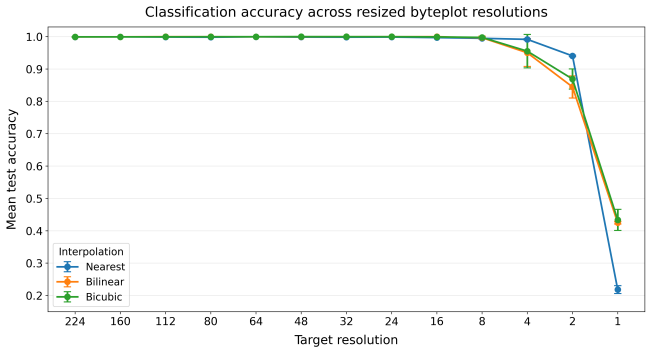


Figure 1: Mean test accuracy across three SHA-grouped folds. All interpolation methods remain near-perfect through 8×8. Nearest-neighbour performs best at 4×4 and 2×2, whereas bilinear and bicubic perform better at 1×1. Error bars show standard deviation across folds.

The interpolation methods separate below 8×8. At 4×4, nearest-neighbour achieves 0.992 accuracy, compared with 0.950 for bilinear and 0.955 for bicubic interpolation. At 2×2, nearest-neighbour remains strongest at 0.940, while bilinear and bicubic decrease to 0.845 and 0.869, respectively. These results suggest that, while multiple spatial positions remain available, retaining selected source values is more useful than replacing them with locally averaged values.

At 1×1, the ordering reverses unexpectedly. Bilinear and bicubic reach 0.425 and 0.434, while nearest-neighbour decreases to 0.218. Once all spatial structure is removed, the smoothing-based methods may provide a more useful global intensity summary because they combine information from a wider region of the byteplot. This result shows that nearest-neighbour is not uniformly preferable: its advantage is limited to small representations that still retain several spatial positions.

5.2 Contrast-loss control

Figure 2 shows that **Gaussian smoothing substantially reduces classification performance**, even when broad byteplot layout remains visible. With a blur radius of 0.5, most moderate resolutions remain accurate, but performance decreases to 0.903 at 8×8 and 0.698 at 4×4. At radius 1.0, performance declines across most resolutions, while radii 2.0 and 4.0 reduce accuracy to weak or near-chance levels.

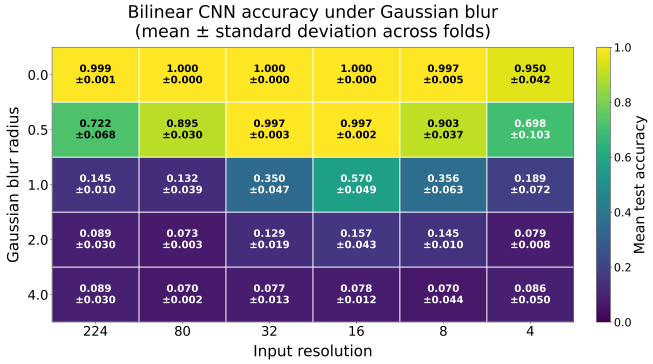


Figure 2: Test accuracy under Gaussian blur. Each cell shows accuracy for one model resolution and blur radius. Performance decreases strongly as blur increases, indicating that local byte-intensity contrast contributes useful signal beyond broad byteplot layout.

These results support the interpretation that local byte-intensity contrast contributes useful information beyond broad layout alone. Notably, this is consistent with the interpolation experiment, where nearest-neighbour outperforms bilinear and bicubic at the smallest multi-pixel resolutions.

The blur trend is not perfectly monotonic across all resolutions. For example, the 80×80 model performs better than some neighbouring resolutions at blur radius 1.0. This is not necessarily contradictory because the same absolute blur radius affects each image size differently, and each resolution-specific model is trained independently. The blur experiment should therefore be interpreted as a contrast-loss sensitivity analysis rather than a strictly resolution-normalised comparison.

5.3 Family-level robustness

Figure 3 shows that **resize robustness is not uniform across malware families or interpolation methods**. The figure reports mean class-wise test accuracy across the three folds for nearest-neighbour, bilinear, and bicubic same-resolution CNN models.

At 8×8, nearly all families remain highly accurate under all three interpolation methods. The differences become clearer at 4×4 and 2×2, where some families remain close to perfect accuracy while others begin to lose family-discriminative information. For example, under bilinear interpolation at 2×2, `BenignELF`, `Botnet`, and `HiddenWasp`

remain perfectly classified, whereas `Trojan`, `Ransomware`, and `Spyware` decrease to 0.51, 0.69, and 0.73, respectively.

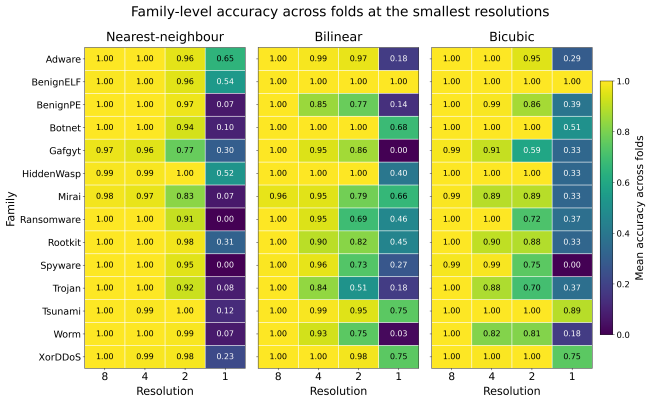


Figure 3: Mean class-wise test accuracy across three folds at the smallest resolutions for nearest-neighbour, bilinear, and bicubic same-resolution CNN models. Family-level robustness differs across both resolution and interpolation method. Differences are most pronounced at 2×2 and 1×1, where several families remain separable under one interpolation method but decline substantially under another.

The largest differences occur at 1×1, where all spatial information is removed. Robustness is strongly interpolation-dependent at this resolution. Under bilinear and bicubic interpolation, `BenignELF`, `Tsunami`, and `XorDDoS` retain relatively high accuracy, whereas several families decline substantially. Under nearest-neighbour interpolation, the ranking changes again: some families that remain strong under smoothing-based interpolation lose accuracy sharply, while others retain more predictive signal.

Overall, high aggregate accuracy at low resolution is not shared equally across classes. Some families remain separable through coarse byteplot regularities or global intensity structure, while others appear to depend more strongly on local contrast, selected byte-derived values, or spatial arrangement. The variation across interpolation methods further indicates that no single set of families should be described as universally robust independently of the preprocessing method.

5.4 Cross-resolution transfer

Figure 4 shows the performance of the 224×224 model when test byteplots are downsampled to a smaller bottleneck resolution and then restored to 224×224 before inference. Error bars show the variation across the three folds.

The high-resolution model loses accuracy as fine spatial information is removed, but nearest-neighbour preserves substantially more transferable signal than bilinear and bicubic. Nearest-neighbour remains above 0.6 accuracy through 64×64 and retains approximately 0.4 accuracy at 48×48. In contrast, bilinear and bicubic decline sharply after 160×160 and remain close to chance level from approximately 48×48 onward.

This result differs from the same-resolution experiment, where all methods remain highly accurate through 8×8. Thus, low-resolution byteplots can still support accurate classification when a model is trained directly on them, but the high-resolution model cannot reliably reuse its learned representation after aggressive information loss. The error bars also show that the broad transfer pattern is consistent across the three folds, although uncertainty increases at some intermediate resolutions.

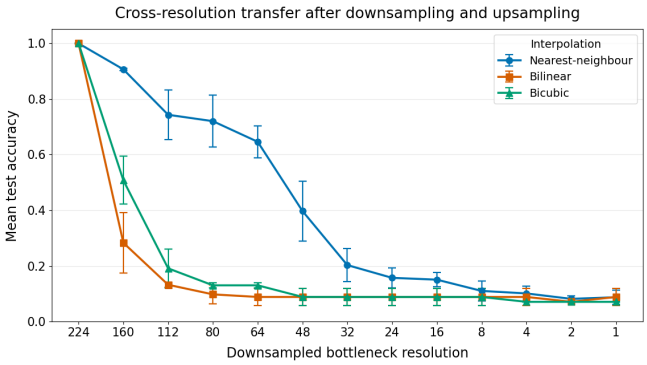


Figure 4: Cross-resolution transfer of the 224×224 CNN. Test byteplots are downsampled to each bottleneck resolution and then restored to 224×224 before inference. Error bars show standard deviation across the three folds.

5.5 Nearest-pixel interpretability

A very small number of nearest-neighbour pixel values retains substantial family-discriminative information. The 4×4 decision tree achieves 0.832 accuracy and 0.843 macro-F1 using only sixteen grayscale values. Surprisingly, the 2×2 representation performs similarly, reaching 0.845 accuracy and 0.842 macro-F1 with only four values.

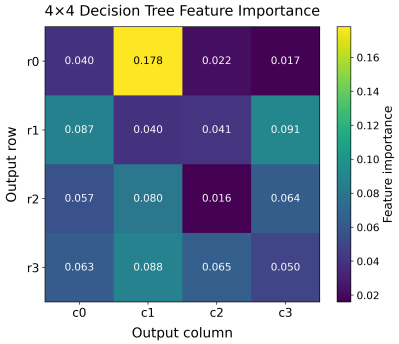


Figure 5: Normalized decision-tree feature importance for the sixteen 4×4 nearest-neighbour pixels. Higher values indicate a larger weighted reduction in Gini impurity.

This small difference should not be interpreted as evidence that four pixels are inherently better than sixteen. Instead, both representations appear to retain sufficient coarse family signal for the diagnostic tree. At 1×1, accuracy decreases to 0.518 and macro-F1 to 0.513. This remains substantially above the random baseline of approximately 0.071 for 14 classes, but it shows that one global intensity value does not preserve enough positional information for reliable family classification.

Figure 5 shows that the 4×4 tree does not assign equal importance to all sixteen pixel positions. A small set of fixed rel-

ative byteplot locations contributes most strongly to the tree decisions, indicating that the retained signal is concentrated rather than evenly distributed across the final image.

The most important 4×4 positions were then mapped back to their original byteplot coordinates and raw-binary offsets. Figure 6 shows that several influential pixels map predominantly to executable code. In particular, `p_r1_c3` and `p_r1_c0` map entirely to code in the analysed samples, while `p_r0_c1` maps predominantly to code with smaller contributions from imports/linkage and other sections. Other important pixels are more mixed: `p_r3_c1` is distributed across imports/linkage, data, and other sections, and `p_r2_c1` spans code, data, and other sections.

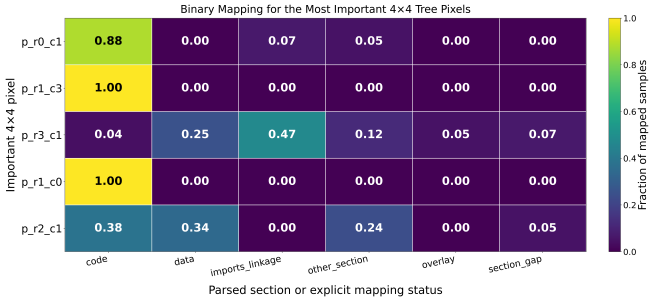


Figure 6: Binary mapping of the most important 4×4 tree pixels. Several important pixels map predominantly to executable code, while others are distributed across multiple parsed binary regions.

These results suggest that the family-discriminative signal retained by nearest-neighbour resizing overlaps meaningful parsed binary regions, especially executable code, rather than arising from completely arbitrary offsets. At the same time, the important pixels do not all map to one universal section. The retained signal appears to come from multiple binary regions, which is consistent with the idea that aggressive resizing preserves coarse family-level structure rather than one uniquely informative semantic component.

6 Discussion

This section interprets the findings according to the four subquestions. Overall, aggressive resizing preserves useful malware-family signal, but this signal is more likely to reflect coarse layout, sampled byte values, local contrast, and family-specific binary regularities than detailed program semantics.

RQ1: How do downsampling and interpolation affect classification accuracy?

Byteplot classification can remain highly accurate after substantial resizing, but the information retained depends on the interpolation method once only a few spatial positions remain. At multi-pixel ultra-low resolutions, nearest-neighbour performs best, whereas bilinear and bicubic become more useful when the image is reduced to a single global value.

Together with the blur experiment, this suggests that broad byteplot layout alone is not sufficient for classification. Preserved local intensity contrast and discrete sampled byte-derived values also contribute useful signal. Nearest-neighbour retains these values directly, whereas bilinear and bicubic replace them with local averages. This provides a plausible explanation for the advantage of nearest-neighbour when several spatial samples remain.

This pattern differs from the result reported by Zhao et al. (2023), where bilinear interpolation performed best in a multi-channel malware-image transfer-learning setting. This should not be interpreted as a direct contradiction because the experimental settings differ substantially. Zhao et al. evaluated interpolation as part of a conventional preprocessing pipeline, whereas the present study progressively compresses single-channel byteplots to extreme resolutions. Under this aggressive-resizing setting, preserving discrete sampled values and local contrast appears more useful than smoothing neighbouring values while spatial structure remains.

RQ2: Which malware families remain robust under resizing?

Low-resolution robustness is not shared equally across malware families and depends on the interpolation method. While most families remain accurately classified at 8×8, their performance diverges at the smallest resolutions. Some retain family-discriminative signal under severe compression, whereas others deteriorate rapidly once local contrast and spatial detail are removed.

The robustness of some families is consistent with the observation that related binaries can produce recurring byteplot structure (Nataraj et al., 2011). However, the precise source of this separability remains uncertain. It may reflect recurring code, resources, or file-layout characteristics, but it may also arise from dataset-specific artefacts such as file size, compiler behaviour, packing conventions, or sample collection. Therefore, low-resolution robustness should be interpreted as evidence of retained family-discriminative regularities, not necessarily robust malware semantics.

RQ3: Can a high-resolution model generalise to low-resolution byteplots?

Strong performance from models trained directly on low-resolution byteplots does not imply that a model trained at 224×224 uses the same evidence. Once test images are compressed and restored to the original input size, the high-resolution models lose accuracy substantially.

Nearest-neighbour preserves more transferable information than bilinear and bicubic at intermediate bottleneck resolutions. This suggests that the high-resolution model benefits more from retained source intensities than from locally averaged reconstructions. Nevertheless, performance declines strongly as the bottleneck becomes smaller, indicating that much of the high-resolution representation depends on spatial detail that does not survive aggressive downsampling. This interpretation is based on observed transfer behaviour rather than a direct measurement of internal feature similarity.

The result aligns with Z. Wang et al. (2016), who found that models trained at different resolutions in natural-image recognition may learn different features. Similarly, the combination of strong directly trained low-resolution models and poor transfer from the 224×224 models suggests that

high- and low-resolution malware classifiers adapt to different forms of byteplot signal.

RQ4: What do nearest-neighbour pixels reveal about the family-discriminative information retained at ultra-low resolutions?

A very small set of nearest-neighbour pixel values retains substantial family-discriminative information. Only a subset of the 4×4 pixels contributes strongly to the fitted tree, indicating that the retained signal is concentrated in particular relative byteplot locations rather than uniformly distributed across the image.

The backmapping analysis further shows that several influential pixels map predominantly to executable code, while others map across data, imports/linkage, and other parsed regions. This supports the byteplot motivation of prior work: recurring binary structure can contribute to family-discriminative visual patterns (Nataraj et al., 2011). It also provides evidence that the retained values are not drawn from completely arbitrary offsets.

However, this does not establish semantic understanding. Nearest-neighbour resizing samples fixed relative positions that are partly determined by resize geometry. A sampled location may overlap with executable code because code sections commonly occur at that relative position, rather than because the model recognises instructions, functions, or malicious behaviour. The strongest conclusion is that low-resolution classification uses byte values sampled from multiple broad binary regions that frequently overlap with executable code, not that the model performs semantic binary analysis.

Limitations and overall interpretation

Several limitations constrain the conclusions. First, the experiments evaluate one dataset and do not test temporal generalisation to newer malware families or future versions of the same families. The observed low-resolution regularities may therefore be specific to the collection period, family definitions, or sample sources represented in the dataset.

Second, compiler settings, packing methods, file size, and other build-related properties may contribute to the observed byteplot structure. Although packed variants were excluded from the main analysis, the unpacked subset may still contain compiler-specific or family-specific layout artefacts. The balanced class distribution is also useful for controlled evaluation but does not reflect real malware prevalence, where family frequencies are often highly uneven.

Third, the backmapping analysis depends on successful parsing of PE and ELF files. Files with missing, malformed, overlapping, or otherwise difficult-to-parse section metadata may be excluded or assigned to broader categories such as overlays or section gaps. The mapped-section proportions should therefore be interpreted as a summary of the parsable samples rather than as a complete description of all binaries.

Fourth, the decision tree is a diagnostic probe rather than an explanation of the CNN. Its feature-importance estimates depend on the selected tree constraints, including maximum depth and minimum samples per split or leaf. Different regularisation settings could alter the relative importance assigned to individual pixels, even if the broad conclusion that a few retained values are predictive remains similar.

Finally, overlap with executable code should be interpreted cautiously. Code sections often occupy substantial or recurrent parts of executable files, so repeated overlap may partly arise from where code tends to occur in the resize geometry. The mapping results therefore show that influential sampled values frequently coincide with meaningful parsed regions, especially executable code, but they do not establish that executable instructions themselves are the causal source of the classification signal.

Overall, aggressive resizing preserves malware-family signal through a combination of coarse byteplot layout, sampled byte-derived values, and local contrast. The relative importance of these signals depends on interpolation, family, and resolution. High low-resolution accuracy therefore demonstrates retained family-discriminative structure, but does not demonstrate preserved detailed code semantics or real-world robustness.

7 Conclusions and Future Work

This paper investigated why malware-image classifiers can remain accurate after aggressive resizing. Classification remained above 0.99 through 8×8 for all studied interpolation methods. At 4×4 and 2×2, nearest-neighbour performed best, indicating that retaining discrete sampled byte values and local contrast is useful while several spatial positions remain. At 1×1, bilinear and bicubic performed better, suggesting that averaging becomes more useful when the image is reduced to one global value.

The blur and cross-resolution experiments show that broad byteplot layout alone does not explain the results. Removing local contrast reduces accuracy, while a model trained at 224×224 does not reliably classify images whose information is reduced before inference, even after they are restored to the original input size. Low-resolution performance therefore appears to require adaptation to the resized representation.

The decision-tree analysis further shows that a few nearest-neighbour pixels retain family-discriminative information. Their mapped locations frequently overlap with parsed binary sections, especially executable code regions, but their positions are also determined partly by resize geometry. The results therefore support an explanation based on coarse binary structure, sampled values, and contrast patterns, rather than semantic understanding of code.

Future work should evaluate these findings on other malware datasets and compare byteplot CNNs with simple baselines such as file size, byte histograms, and entropy features. This would help determine whether the strong ultra-low-resolution results reflect spatial byteplot structure or dataset-specific artefacts.

8 Responsible Research

This section outlines the defensive purpose of the work, the measures taken to support reproducibility, the limited use of AI-assisted tools, and the main threats to the validity and generalisability of the findings. Together, these considerations

clarify both how the study was conducted responsibly and how its conclusions should be interpreted.

8.1 Defensive use of malware data

This project uses malware data solely for defensive research. Its purpose is to understand how static malware-image classifiers behave under aggressive resizing, not to develop, modify, or improve malicious software. The experiments use provided, neutered binaries and derived byteplot representations; no malware is executed.

8.2 Reproducibility and code availability

Reproducibility is supported through fixed random seeds, SHA-256-grouped data splits, and explicit preprocessing settings. The interpolation, blur, cross-resolution, and family-level analyses use three-fold grouped cross-validation, while the decision-tree analysis uses a fixed train/validation/test split as a diagnostic probe. The methodology specifies the interpolation methods, resolution range, training configuration, and decision-tree parameters.

The code used for preprocessing, training, evaluation, visualisation, and pixel-to-binary mapping is available in the accompanying GitHub repository: https://github.com/criss2507/malware-byteplot-resizing. The repository contains the implementation code, configuration files, and derived outputs required to reproduce the reported analyses, but does not redistribute the underlying malware dataset, raw binaries, or restricted dataset material. The notebooks also save intermediate outputs, including metrics tables and pixel-to-binary mappings, so that the main observations can be checked.

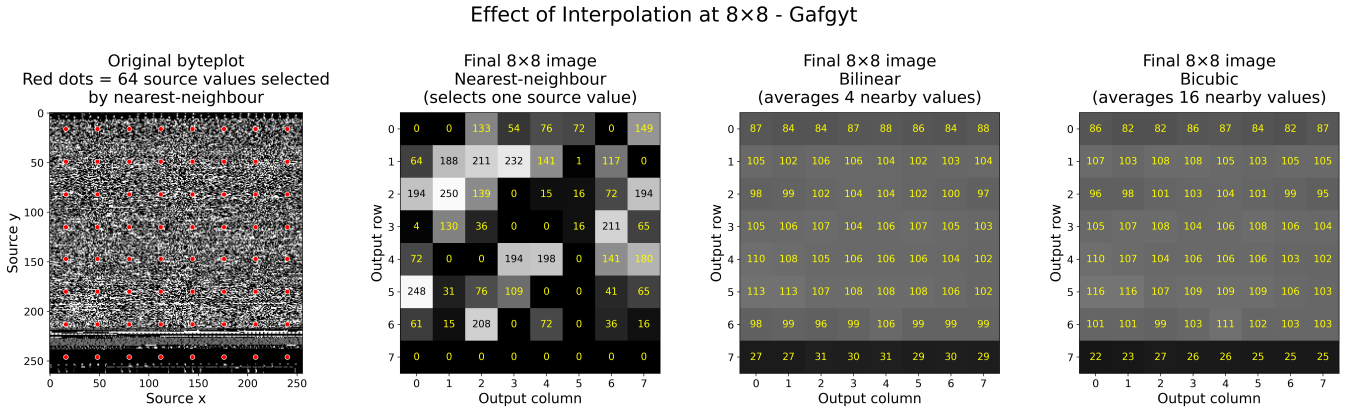
8.3 Use of AI-assisted tools

AI-assisted tools were used to support language editing of the report and to review or debug parts of the implementation. Their use was limited to assistance with wording, code-review suggestions, and troubleshooting. All experimental design decisions, data processing, model training, result selection, interpretation, and final verification were performed by the author. AI-generated suggestions were reviewed critically and were not treated as evidence or as a substitute for scientific judgement.

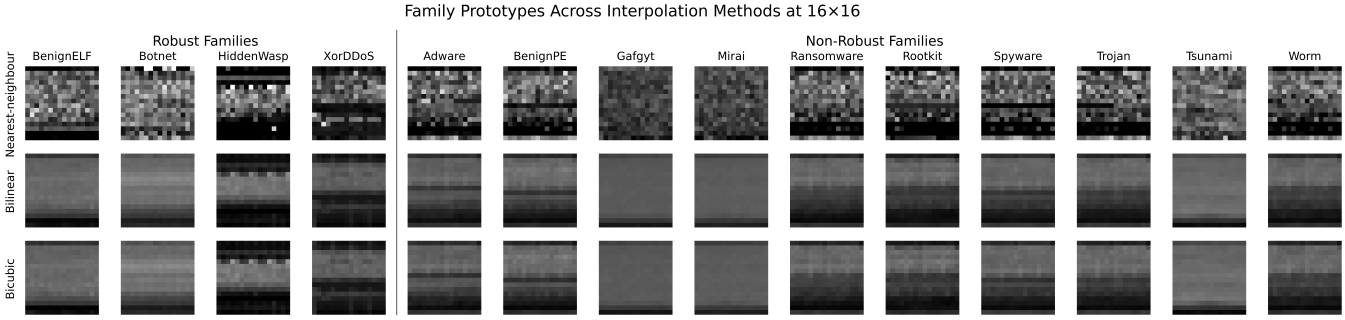
8.4 Scope and limitations

The conclusions are limited to the dataset and experimental setup used here. The dataset may contain artefacts that are not representative of malware in the wild, and high accuracy at very low resolutions may reflect family-specific layout shortcuts, file-size effects, packing conventions, or collection bias. This concern is particularly relevant at 4×4, where only sixteen sampled pixels are available while classification accuracy remains extremely high. The paper therefore does not make claims about real-world deployment performance or semantic understanding of malicious code.

A Visual Effect of Interpolation



(a) Interpolation of one Gafgyt byteplot to 8×8. Red markers indicate the source positions retained by nearest-neighbour interpolation.



(b) Average 16×16 byteplot prototypes for robust and non-robust families. Robust families achieve 100% bilinear classification accuracy at every tested resolution from 224×224 to 4×4.

Figure 7: Visual effect of nearest-neighbour, bilinear, and bicubic interpolation. Nearest-neighbour preserves selected source intensities and local contrast, whereas bilinear and bicubic produce smoother spatial averages. The family prototypes illustrate how smoothing can reduce visually distinct local patterns, particularly among families whose classification accuracy declines at low resolutions.

A.1 Interpolation of an individual byteplot

Figure 7(a) illustrates how one Gafgyt byteplot is reduced to an 8×8 representation. The red dots in the original byteplot indicate the 64 source positions selected by nearest-neighbour interpolation. Each pixel in the final nearest-neighbour image therefore corresponds directly to one retained byte-derived intensity value.

Nearest-neighbour preserves strong local variation, including dark, bright, and intermediate values at fixed relative positions. Bilinear interpolation combines nearby source values and produces a smoother representation with more similar grey levels. Bicubic interpolation also smooths the image through a larger weighted neighbourhood. In this example, both smoothing-based methods preserve the broad dark lower region of the byteplot but remove much of the local contrast retained by nearest-neighbour interpolation.

A.2 Family-level visual comparison

Figure 7(b) compares representative family prototypes at 16×16. The robust group contains the families that achieved 100% classification accuracy with bilinear interpolation across all resolutions, down to 4×4.

The visual comparison shows that nearest-neighbour generally retains more family-specific local patterns, including isolated bright values, sharp horizontal boundaries, and block-like changes in intensity. In contrast, bilinear and bicubic interpolation preserve broader bands and gradual transitions, but suppress much of this local variation.

This difference is particularly visible among the less robust families in Figure 7(b). Under bilinear and bicubic interpolation, several family prototypes become visually similar: they are dominated by comparable smooth grey regions, broad dark lower bands, and weak horizontal transitions. The nearest-neighbour versions remain more distinct from one another because selected source intensities and sharper local contrast are retained.

This observation is consistent with the main interpolation results, where nearest-neighbour performs better once only a small number of output pixels remains. However, the figure is illustrative rather than causal evidence. Visual similarity between prototypes does not by itself establish why the CNN confuses particular families, nor does it identify which exact byteplot structures determine a prediction. It instead provides an intuitive explanation of how smoothing can reduce the visibly distinct information available after aggressive resizing.

References

Aslan, O., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access*, *8*, 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth.

Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G.-g., & Chen, J. (2018). Detection of malicious code variants based on deep learning. *IEEE Transactions on Industrial Informatics*, *14*(7), 3187–3196. https://doi.org/10.1109/TII.2018.2822680

Dai, Y., Li, H., Qian, Y., & Lu, X. (2018). A malware classification method based on memory dump grayscale image. *Digital Investigation*, *27*, 30–37. https://doi.org/10.1016/j.diin.2018.09.006

Freitas, S., Duggal, R., & Chau, D. H. (2022). MalNet: A large-scale image database of malicious software. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 3948–3952. https://doi.org/10.1145/3511808.3557533

Fu, J., Xue, J., Wang, Y., Liu, Z., & Shan, C. (2018). Malware visualization for fine-grained classification. *IEEE Access*, *6*, 14510–14523. https://doi.org/10.1109/ACCESS.2018.2805301

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, *153*, 102526. https://doi.org/10.1016/j.jnca.2019.102526

Kiger, J., Ho, S. S., & Heydari, V. (2022). Malware binary image classification using convolutional neural networks. *International Conference on Cyber Warfare and Security*, *17*, 469–478. https://doi.org/10.34190/iccws.17.1.59

Kumar, S., & Janet, B. (2022). DTMIC: Deep transfer learning for malware image classification. *Journal of Information Security and Applications*, *64*, 103063. https://doi.org/10.1016/j.jisa.2021.103063

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*, 4765–4774. https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767 - Abstract .html

Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011). Malware images: Visualization and automatic classification. *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, 1–7. https://doi.org/10.1145/2016904.2016908

PyTorch Contributors. (n.d.). ResNet18. https://docs.pytorch.org/vision/stable/models/generated/torchvision.models.resnet18.html

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Selinger, M. (2026, March). AV-TEST awards 2025: Celebrating the very best of IT security products. https://www.av-test.org/en/news/av-test-awards-2025-celebrating-the-very-best-of-it-security-products/

Sihwail, R., Omar, K., & Zainol Ariffin, K. A. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *International Journal on Advanced Science, Engineering and Information Technology*, *8*(4-2), 1662–1671. https://doi.org/10.18517/ijaseit.8.4-2.6827

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

Verma, V., Muttoo, S. K., & Singh, V. B. (2020). Multiclass malware classification via first- and second-order texture statistics. *Computers & Security*, *97*, 101895. https://doi.org/10.1016/j.cose.2020.101895

Wang, C., Zhao, Z., Wang, F., & Li, Q. (2021). A novel malware detection and family classification scheme for IoT based on DEAM and DenseNet. *Security and Communication Networks*, *2021*, 1–16. https://doi.org/10.1155/2021/6658842

Wang, Z., Chang, S., Yang, Y., Liu, D., & Huang, T. S. (2016). Studying very low resolution recognition using deep networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 4792–4800. https://doi.org/10.1109/CVPR.2016.518

Zhao, Z., Yang, S., & Zhao, D. (2023). A new framework for visual classification of multi-channel malware based on transfer learning. *Applied Sciences*, *13*(4), 2484. https://doi.org/10.3390/app13042484