An analysis of different xAI methods for explaining malware image classifications

Bram de Jonge[1]

Supervisor(s): Tom Viering[1]**, Akash Amalan**[1]

[1]**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: Bram de Jonge
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Akash Amalan, Georgios Smaragdakis

Abstract

Malware image classification using CNNs has achieved classification accuracies of up to 99%, but the black-box nature of these models limits their trustworthiness. Explainable AI (xAI) methods can highlight which image regions drive a model's predictions, yet prior work has not mapped these regions back to named binary sections or quantitatively validated their importance. This paper addresses both gaps by applying Grad-CAM, LIME, and SHAP to a ResNet-18 classifier. For each test sample, highlighted image regions are mapped to named binary sections, and an occlusion experiment validates that high-attention sections are genuinely critical to classification. All three methods consistently rank `.rdata`, `.data`, and `.eh_frame` as the most important sections for PE binary classification, with attention distributed more broadly across ELF binaries. Occluding the top three sections per family causes accuracy to drop below 50% across all methods, confirming the importance of the identified regions. SHAP identifies the single most important section most precisely, while Grad-CAM achieves comparable overall performance at 50–60 times the speed. Notably, high classifier attention on `.eh_frame` suggests that the model may exploit tool-specific artifacts, raising concerns that should be accounted for before deploying these classifiers in production.

1 Introduction

Malware remains one of the most persistent threats in modern cybersecurity. The AV-TEST Institute registers over 450,000 new malicious programs daily, with the total number of known samples surpassing 1.3 billion in 2023 [1]. Automated malware detection systems are tasked with keeping up with these increasing numbers, but standard malware analysis techniques struggle in multiple aspects, such as speed and their ability to identify mutated samples.

A promising new direction for malware analysis emerged in 2011, when Nataraj et al. proposed the use of malware images, grayscale images with pixels corresponding to the raw byte values of the corresponding malware sample [2]. They showed that these images contain family specific patterns, achieving a 97% classification accuracy using GIST features and k-nearest neighbors. Subsequent work made use of convolutional neural networks (CNN) to improve on this accuracy, achieving classification accuracies of 98-99% on unpacked malware, and a notable 98% accuracy on packed malware [3; 4]. These high accuracies are a solid argument for malware image based classification being able to compete with more established methods of static or dynamic analysis, often at a much greater speed.

However, a major issue of deep learning methods such as CNNs is that they are effectively a black box model. It is known that they perform well, but it is not known *why* they perform so well. In domains such as cybersecurity, where a small mistake can lead to catastrophic results, being able to trust these models is of great importance. If a classifier cannot be understood, it cannot be trusted, limiting its actual usefulness in the field. Especially with these high accuracies, a solid explanation is needed to ensure the classifier learns actually meaningful things, and cannot be easily circumvented by adversaries.

Explainable AI (xAI) is a rapidly developing field with the goal to address this issue, by providing insight into what the model deems important in its decision making process. Gradient-weighted Class Activation Mapping (Grad-CAM) [5], Local Interpretable Model-agnostic Explanations (LIME) [6], and SHapley Additive exPlanations (SHAP) [7] are among the most widely used techniques for image classification tasks, each highlighting which regions of an input image most influence a prediction. Prior work has applied these methods to malware image based classifiers [8; 9; 10], but these studies often only analyze their visual results and only map back to the original file structure for a few samples, if any at all. As a result, it remains unclear where the highlighted regions actually point to, and what these regions represent. Furthermore, no prior work has quantitatively compared the effectiveness of different xAI methods for malware image classification or validated whether their identified regions are actually important for classification.

This paper addresses these gaps through an analysis of Grad-CAM, LIME and SHAP applied to a CNN trained to classify malware byteplot images into families. For each sample in the test set, regions highlighted by xAI methods are mapped back to named sections of the original malware binaries, allowing for a quantitative analysis of which binary sections each xAI method deems important. These findings are then validated by occluding the highlighted sections, and comparing the drop in accuracy when reevaluating the classifier. These occlusion results are then also used to compare the effectiveness of each xAI method.

This analysis is guided by the following research question:

> *Which Explainable AI method is most effective for explaining CNN decisions on malware byteplot images, and how do the methods differ in their explanations?*

Which is further divided into four subquestions:

1. Which file sections receive the most attention, according to xAI techniques?
2. When occluding high attention sections found by xAI, does accuracy indeed drop?
3. How do the different xAI techniques compare in terms of effectiveness?
4. What are the differences between the sections highlighted by different xAI methods?

The remainder of this paper is structured as follows. Section 2 reviews related work on malware image classification and explainable AI. Section 3 describes the dataset, model architecture, xAI pipeline, and section mapping procedure. Section 4 presents the results for each subquestion. Section 5

interprets these findings in relation to prior work and discusses limitations. Section 6 summarises the main conclusions and outlines directions for future work.

2 Background and Related Work

2.1 Malware Images

Malware images were first introduced by Nataraj et al. [2], who converted malware binaries into grayscale images with fixed widths. They showed malware of the same family produces similar visual patterns, and reached a classification accuracy of 97% using GIST features and k-NN. This representation enables the application of computer vision techniques to malware classification. Subsequent studies [3; 4] then made use of deep learning methods, most notably Convolutional Neural Networks (CNNs), to reach even higher classification accuracies. CNN-based approaches consistently achieved classification accuracies of 98–99%, establishing malware image classification as a promising alternative to more traditional malware analysis methods. However, it remains unclear what features these models actually rely on.

2.2 Explainable AI

Explainable AI (xAI) aims to provide insight into the decision making process of complex models such as CNNs. As deep learning models become increasingly complex, interpretability methods are important for understanding which parts of the input contribute most to model predictions. Several explainability techniques have been proposed for image classification tasks, each offering different perspectives on model behaviour.

Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) [5] is a gradient-based explainability method designed for CNNs. Grad-CAM computes the gradients of a target class with respect to the feature maps of a convolutional layer, and uses these gradients to generate a heatmap indicating which regions of the input image contributed most strongly to the prediction. As this only requires a single backwards pass per sample, Grad-CAM is highly efficient when compared to other xAI methods. However, it does require access to the internals of a classification model, meaning it cannot always be applied.

LIME

Local Interpretable Model-agnostic Explanations (LIME) [6] is a model-agnostic explainability technique that approximates the local decision boundary of a complex model using a simpler interpretable model. For image classification tasks, LIME mutates segments of the input image and measures how these mutations influence the prediction to build a simpler model, such as a decision tree. Based on this model, it identifies which regions contribute positively or negatively to a classification decision using the decision boundaries of the model. Unlike Grad-CAM, which relies on internal gradients of CNNs, LIME can be applied to any classifier. However, LIME requires multiple evaluations per sample, making it much slower than Grad-CAM

SHAP

SHapley Additive exPlanations (SHAP) [7] is an explainability framework based on Shapley values from cooperative game theory. SHAP estimates the contribution of individual input features to a model prediction by measuring their marginal impact across different feature combinations. As true per-pixel SHAP values would be computationally infeasible to calculate, combinations of different image regions instead of pixels are tested to see how much each image region contributes to each output. As with LIME, SHAP also requires multiple evaluations of the model per sample, hindering its speed.

Validation through occlusion

In addition to explainability techniques themselves, occlusion can be used as a validation method to verify whether highlighted regions are genuinely important for model predictions. Occlusion involves masking selected regions of the input image and observing the effect on classification confidence or accuracy [11]. If occluding regions identified by methods such as Grad-CAM, LIME, or SHAP leads to a significant decrease in prediction confidence or classification accuracy, this provides evidence that these regions are indeed important for the model's decision making process.

2.3 xAI applied to Malware Image Classification

Several studies have applied explainability methods to CNN-based malware image classifiers, but they share important limitations that this paper aims to address.

Iadarola et al. [8] were among the first to couple malware image classification with a gradient-based explainability method. They applied Grad-CAM to a CNN trained on Android APK grayscale images, using the resulting heatmaps to direct the attention of security analysts toward the most important regions of each sample. The stated goal was reducing the amount of code an analyst must manually inspect, with explainability serving as a triage tool rather than as a subject of study in itself. Importantly, the analysis remains qualitative, as highlighted regions are visually inspected but never mapped back to named sections of the underlying binary, and no quantitative comparison against a baseline or between alternative xAI methods is performed.

Brosolo et al. [9] take a broader view, conducting replicability experiments across three datasets and applying Grad-CAM and a related method called HiResCAM to study what CNN malware classifiers attend to. A key contribution of their work is using xAI outputs not merely for explanation but to improve classification: they introduce an image-masking technique using CAM heatmaps, achieving F1-score gains of 2–8% across datasets. However, as in prior work, the explanations remain at the level of image regions. The authors do not link highlighted areas back to specific binary structures such as code sections or resource tables, so it is impossible to determine whether the model learns from meaningful parts of the executable.

Nazim et al. [10] present the most direct predecessor to the present work, applying all three methods studied here to a CNN ensemble trained on a blended malware image

dataset. They provide both global feature importance analysis and per-sample visualizations, and discuss which methods produce more informative heatmaps from a qualitative standpoint. Nevertheless, the study does not relate highlighted regions to binary section names beyond a few high-level examples, does not quantitatively validate whether highlighted regions are actually important to the classifier through occlusion, and does not offer a quantitative comparison of method effectiveness.

Taken together, these studies establish that xAI methods can produce visually informative heatmaps for malware image classifiers, but they leave two central questions unanswered. First, none map highlighted image regions back to the actual binary, so it remains unclear whether the model's attention corresponds to meaningful executable structures. Second, only Brosolo et al. validates their explanations quantitatively, and for a different purpose. This paper addresses both gaps through a systematic section-mapping procedure and a controlled occlusion experiment applied across all three xAI methods.

3 Methodology

This section describes the dataset, image representation, classifier, xAI techniques and procedures to map regions back to code and validate through occlusion. The overall pipeline consists of four stages: (1) training a baseline classifier, (2) applying xAI methods to a test set, (3) mapping back the highlighted regions to code sections of the original samples, and (4) validating whether these regions are important through occlusion.

3.1 Dataset

The experiments use a dataset of 10,010 malware samples provided by Akash Amalan. The dataset spans 14 families, comprising of both PE (Portable Executable) and ELF (Executable and Linkable Format) binaries. The dataset is evenly split up into 715 samples per malware family. Details on the different families can be found in table 1. The dataset does not contain any functional malware samples, instead consisting of neutered samples that have the same signatures as functional samples.

3.2 Image Representation

Following the approach introduced by Nataraj et al. [2], each binary file is converted into a grayscale image by interpreting the raw bytes as 8-bit pixel values in the range $[0, 255]$. The byte sequence is arranged row by row into a two-dimensional matrix with a width and height depending on the file's size. The width is chosen according to the table of recommended widths for certain file sizes from Nataraj et al.'s work, and the height is equal to the file size in bytes divided by the width. All images are then resized bilinearly to 224×224 pixels to satisfy the input requirements of the ResNet-18 architecture used for the classifier. Images are then finally normalized using the means and standard deviations from the ImageNet dataset [12].

Family	Description
Adware	Displays unwanted advertisements
BenignELF	Non-malicious ELF files
BenignPE	Non-malicious PE files
Botnet	Controls device to be part of remote-controlled network
Gafgyt	Linux based botnet malware for IoT devices
HiddenWasp	Linux based trojan family
Mirai	IoT botnet malware using password guessing
Ransomware	Encrypts device and asks for payment in exchange for decryption
Rootkit	Gains privileged access by modifying system components
Spyware	Collects sensitive information such as passwords
Trojan	Malware disguised as legitimate software
Tsunami	Linux based botnet malware for IoT devices
Worm	Self-replicates without requiring user interaction
XorDDoS	Linux based botnet malware using XOR-based obfuscation

Table 1: Descriptions of Malware Families

3.3 Classifier

A ResNet-18 convolutional neural network [13] is trained on the dataset as the baseline classifier. ResNet-18 has a good balance of classification performance and computational efficiency, and has already been used previously with the three xAI methods used in the experiments. The model is trained for 30 epochs using the Adam optimizer, with a learning rate of 0.0001, batch size of 128 and with cross-entropy loss over the 14 output classes. Training is done using the PyTorch framework on a RTX 4070 GPU. The dataset is split up into a 70-15-15 train-val-test split stratified by family, with the test split being saved for use in the subsequent experiments as well.

3.4 Explainability Methods

Three explainability methods are applied to the trained classifier: Grad-CAM, LIME and SHAP. Each of these methods produces a heatmap, showing the most important areas of an image for the model's prediction. The methods are applied to every sample in the test set, resulting in a heatmap per sample per method.

Grad-CAM

Grad-CAM analysis is done using the pytorch-grad-cam python library [14]. While this library allows for multiple different variations of CAM based methods, the original implementation of Grad-CAM was chosen as a representative method. In this study, Grad-CAM is applied to the final convolutional layer of ResNet-18, which is the most widely used layer in Grad-CAM studies.

LIME
LIME analysis is done using the LIME python library [15]. The image input is first segmented into superpixels using the quickshift algorithm [16] with a kernel size of 4. These are then mutated to create 500 new samples close to the original sample. Each of these samples is then passed through the classifier resulting in new outputs. A linear model is then trained on this new data, and the coefficients of this model assign either a positive or negative importance value to each superpixel. Positively contributing superpixels are then used to indicate the important regions of the image according to LIME.

SHAP
SHAP analysis is done using the SHAP python library [17]. The automatic algorithm selection of the library was used, which picked PartitionExplainer as the most suitable algorithm based on the model. SHAP values are estimated by masking image regions using the inpainting technique described by A. Telea [18]. After 100 different combinations of maskings, a heatmap is then constructed by aggregating the positively attributing SHAP values for the target output.

3.5 Section Mapping

A crucial part of this study is the mapping of image attention regions to code sections of the corresponding binary file. This is done by linking back the rows of the image to the binary regions in the original file. While doing this per pixel is also an option, this would lead to less accurate results, as the resizing to and from 224×224 pixels results in lost information and one pixel being linked back to multiple bytes. Therefore, a per row approach is used to get a rough estimation.

The xAI heatmap, which is initially produced at a 224×224 resolution, is resized to the original byteplot image's $H \times W$ using bilinear interpolation. A row-level attention profile is then computed by averaging the heatmap values across each row, resulting in a one-dimensional vector $a \in \mathbb{R}^H$ where a_r represents the mean attention assigned to row r.

The section layout of each binary is parsed, resulting in names, offsets and sizes of each named section of a file. These sections contain specific kinds of info, such as .text containing program logic, and .idata containing import tables. The offsets and sizes are then converted to row indices. A section spanning offsets $[off, off + size)$ is mapped to rows $\lfloor off/W \rfloor$ through $\lfloor (off + size)/W \rfloor$. The final attention score of a section is then the mean of a_r over all rows within this range. For PE files, the header is also counted as a separate section called PE_HEADER.

Finally, section attention scores are aggregated across samples at the family level by computing the mean score per section per family. This results in a $method \times family \times section$ matrix showing the attention profiles according to each method. Because PE and ELF binaries have different section names, they are analyzed separately to avoid a sparse matrix.

3.6 Occlusion Validation

To validate that sections identified as high-attention by each xAI method are indeed important to the model's predictions, an occlusion experiment is conducted. For each method and each malware family, the most important sections according to average xAI attention are occluded to measure the effect on the classifier's accuracy.

Occlusion is done by zeroing all bytes within the targeted image region, resulting in black pixels. To ensure that this particular method of occluding leads to reasonable results, another experiment was done by occluding with random noise for comparison. After occluding the top k sections (for $k = 1, 2, 3, 4, 5$), the images are passed through the classifier and accuracy is recorded on the modified test set.

A random baseline is established by assigning sections randomized attention scores, and repeating the above procedure 5 times, taking the average accuracy of the 5 runs. A method is considered to be able to identify regions important to the classifier if occluding its important sections results in a steeper accuracy drop than the random baseline.

To ensure that the bottom ranked sections for each xAI method are also unimportant to the classifier, the reverse experiment is also done, with bottom sections being occluded. In this case, a xAI method performs well if accuracy degrades slower than the random baseline.

4 Results

This section presents results for each of the four subquestions: the sections most attended by each xAI method (SQ1), the effect of occluding those sections on classifier accuracy (SQ2), a comparison of method effectiveness (SQ3), and the differences between methods (SQ4).

4.1 SQ1: Which file sections receive the most attention, according to xAI techniques?

Table 2: Global average section attention scores per method (top 10 by Grad-CAM rank) for PE samples.

Section	Grad-CAM	LIME	SHAP
`.rdata`	**0.692**	**0.619**	**0.420**
`.eh_frame`	0.685	0.357	0.389
`.data`	0.634	0.532	0.410
`.edata`	0.628	0.199	0.279
`.idata`	0.527	0.246	0.246
`.CRT`	0.417	0.245	0.191
`.text`	0.389	0.244	0.317
`.tls`	0.389	0.247	0.190
`.reloc`	0.385	0.261	0.190
`.bss`	0.111	0.077	0.245

Tables 2 and 3 show the average attention scores per method across all malware samples for PE and ELF samples. There are 11 unique sections for PE binaries, and 32 unique sections for ELF binaries. For PE binaries, all three methods agree on `.rdata`, `.eh_frame` and `.data` being the most important sections for classification, with `.rdata` being ranked the highest across every method. For ELF binaries, attention is distributed more evenly across many sections, with 11 sections receiving a Grad-CAM attention above 0.5. SHAP noticeably deviates quite considerably from the other methods.

Table 3: Global average section attention scores per method (top 10 by Grad-CAM rank) for ELF samples.

Section	Grad-CAM	LIME	SHAP
.fini	**0.572**	0.330	0.237
.rodata	0.548	**0.340**	0.251
.fini_array	0.535	0.298	0.159
.init_array	0.535	0.298	0.159
.got	0.534	0.301	0.159
.data	0.531	0.300	0.159
.bss	0.528	0.207	0.184
.shstrtab	0.527	0.275	0.159
.text	0.525	0.278	**0.338**
.eh_frame	0.521	0.328	0.224

Appendix A shows the full breakdown of the attention scores per section per family per method.

4.2 SQ2:When occluding high attention sections found by xAI, does accuracy indeed drop?

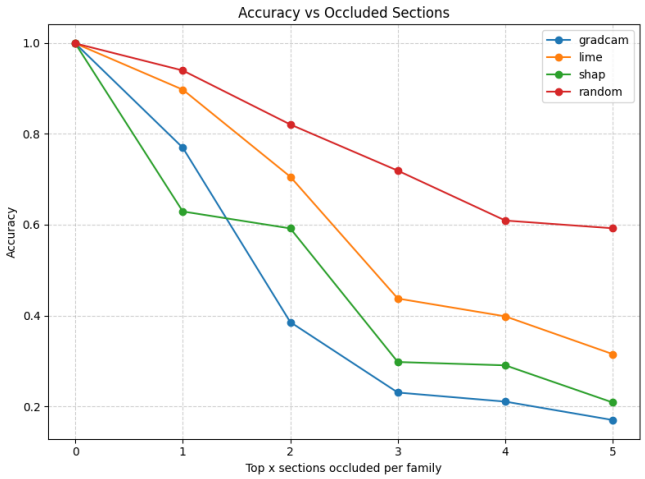

Figure 1: Accuracy drop after occluding the top k sections for each method.

The occlusion experiment confirms that the sections identified as important by all three xAI methods are genuinely critical for the classifier's predictions. Figure 1 reports accuracy as a function of the number of top-ranked sections occluded per family, alongside the random baseline averaged over five runs. The clean accuracy without any occlusion is 99.93%.

All three methods result in accuracy drops that exceed the random baseline across all values of k, and after occluding the top 3 sections all methods result in an accuracy below 50%.

For the reverse experiment of occluding the least important sections, results are less consistent, as shown in figure 2. For $k = 2$ all methods are able to maintain an accuracy above 99.8%.

These results seem to stay consistent regardless of the occlusion method used, with the same experiment using noise occlusion showing similar results, as shown in appendix B.

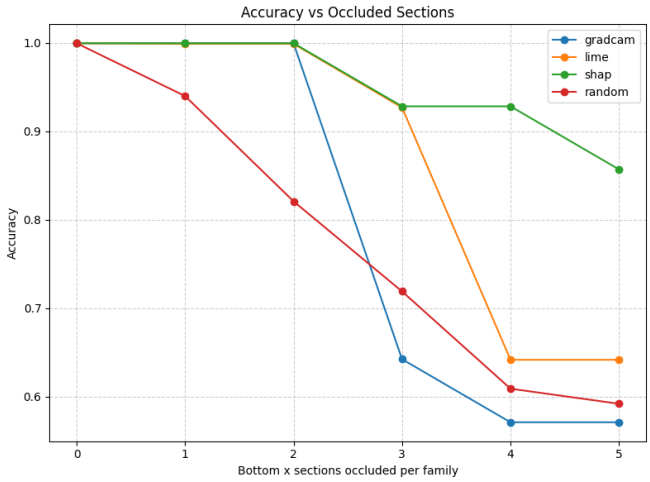

Figure 2: Accuracy drop after occluding the bottom k sections for each method.

4.3 SQ3: How do the different xAI techniques compare in terms of effectiveness?

Comparing the curves in figure 1 shows that SHAP identifies the most important (top 1) sections most efficiently. At $k = 1$, SHAP results in the largest accuracy drop (99.93% to 62.92%). At $k = 2$ however, Grad-CAM achieves an even bigger drop (76.96% to 38.55%). From $k = 3$ and onwards, Grad-CAM and SHAP converge to have roughly the same performance. LIME performs the worst, having the highest accuracy of the three for all k.

In the bottom section occlusion experiment, all three methods maintain roughly the same accuracy up to $k = 2$. Grad-CAM has a large drop in accuracy to 64.3% at $k = 3$, which is roughly 8% lower than the random baseline accuracy. LIME experiences a similar drop at $k = 4$, nearly diving below the random baseline as well, while SHAP is able to maintain a reasonably high accuracy of 85.7% at $k = 5$.

Table 4: Pearson Correlation Coefficients between section sizes and section attention scores for the three xAI methods, and p-values indicating the probability of these results if there is no correlation.

Method	ρ	p-value
Grad-CAM	0.097	7.05×10^{-49}
LIME	0.012	7.78×10^{-2}
SHAP	0.125	1.15×10^{-80}

Since it is possible that some xAI methods may just be highlighting larger sections, resulting in larger sections being occluded, section attentions were compared with section sizes for each xAI method. The results of this can be found in table 4. Grad-CAM and SHAP do show strong evidence for there being correlation ($p < 0.05$), but their coefficients show there is only some very weak correlation. LIME does not show enough evidence for any correlation.

When comparing runtimes of the different methods, Grad-CAM took *35s* to calculate the attention across all test samples, SHAP took *27m23s*, and LIME took *31m15s*.

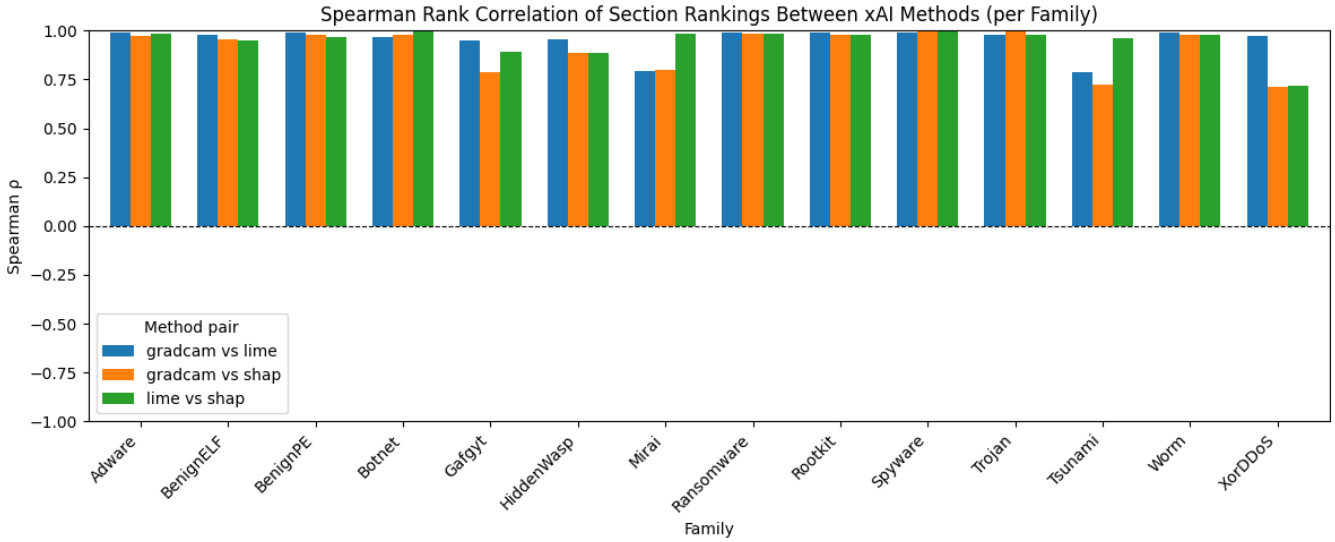

Figure 3: Spearman Rank Correlation between methods across different families. Values range from -1 to 1, with numbers closer to 1 indicating more correlation.

4.4 SQ4: What are the differences between the sections highlighted by different xAI methods?

Spearman rank correlations reveal high agreement between the three xAI methods. Grad-CAM and LIME show the highest mean agreement ($\rho = 0.951$, $\sigma = 0.070$), followed closely by LIME and SHAP ($\rho = 0.947$, $\sigma = 0.074$), with Grad-CAM and SHAP having the lowest ($\rho = 0.909$, $\sigma = 0.106$). Correlations per family can be found in figure 3. This figure shows that while correlation is high for most families, some ELF families such as XorDDoS have one pair with high correlation, while the other 2 pairs go down to roughly 0.75.

Despite this high correlation, some major differences between methods can still be found when comparing specific sections, as shown in figures 4 and 5.

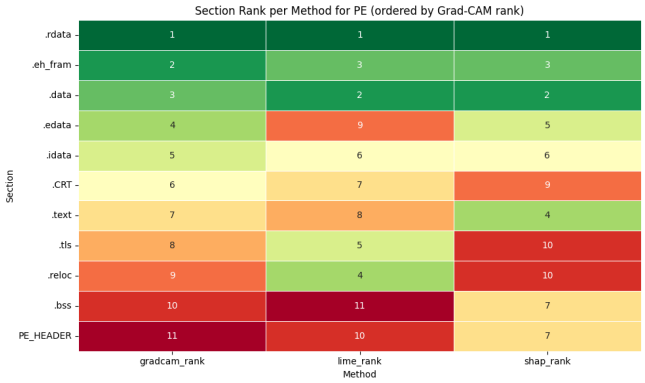

Figure 4: Global section rankings per method for PE files

For PE binaries, Grad-CAM and SHAP weigh `.edata` considerably more than LIME. On the other hand, LIME weighs `.tls` and `.reloc` a lot more than either Grad-CAM or SHAP. SHAP also weighs `.text` somewhat higher than the other methods.

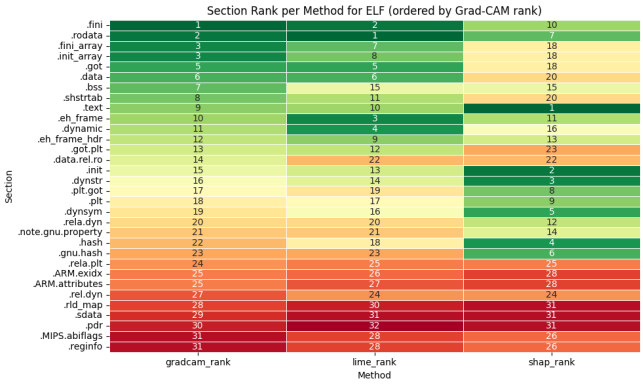

Figure 5: Global section rankings per method for ELF files

For ELF binaries, SHAP turns out to be very different from Grad-CAM or LIME, with its rankings wildly differing from the rest. When comparing Grad-CAM and SHAP, only 3 of the top 10 sections from Grad-CAM, are in the top 10 for SHAP. SHAP also once again highlights `.text` more strongly than the other methods, similarly to its results in PE binaries. LIME and Grad-CAM maintain roughly the same ordering, with `.eh_frame` and `.dynamic` being the main standouts for LIME, and `.bss` being the main standout for Grad-CAM. The three methods do agree on their least important sections, with the bottom 9 being the same for all three.

5 Discussion

5.1 Interpretation of Section Attention

The finding that all three xAI methods consistently rank `.rdata` as the most important section for PE binary clas-

sification is notable and interpretable. The `.rdata` section contains read-only data such as string literals and constants, which is information that can be highly family-specific. For example, Ransomware often contains certain strings meant to inform the user that their device has been encrypted, and that they should send money through a certain method. Similarly, `.data` being in the top 3 also makes sense, as this also contains initialized data relevant to file behavior. The high attention on `.eh_frame` is more surprising, as this section stores exception handling metadata that does not reflect the file's behavior. Sections such as `.eh_frame` are almost always mechanically generated by the compiler, and therefore mostly reflect the tools used to write the program as opposed to the actual behavior of the program. This may indicate the classifier found a way to narrow down the possible families based on the compiler used. This is not reliable however, as adversaries could easily strip this section or change what tools they use.

For ELF binaries, the broader distribution of attention across many sections is consistent with ELF malware in this dataset spanning a more diverse range of families, including several IoT-targeted botnets (Gafgyt, Mirai, Tsunami, XorDDoS). These families may share large portions of their binary structure while differing in relatively small regions, causing xAI methods to distribute attention more broadly rather than concentrating on one or two dominant sections. Many ELF sections also only consist of just a couple of bytes, which is information that may be lost when the images are being resized to fit the CNN input.

SHAP's divergence from Grad-CAM and LIME on ELF binaries warrants further investigation. A plausible explanation is that SHAP focuses more on local features compared to Grad-CAM and LIME, with it assigning values to smaller sections of images. In contrast, LIME makes use of larger superpixels and Grad-CAM is applied to the final convolutional layer, which also looks at more abstract, larger image regions.

The `.text` section consistently being ranked quite low by Grad-CAM and LIME is also notable, as this is the section that contains the actual code logic for the file. While intuitively this would be the section that is most informative when it comes to malicious behavior, Grad-CAM and LIME show that this section produces less recognizable patterns compared to other sections. That said, the higher SHAP attention does indicate there might still be some local features that are useful to the classifier.

5.2 Occlusion Validation

The occlusion results strongly confirm that the sections identified as important by all three methods are genuinely critical to the classifier. The drop to below 50% accuracy after occluding only the top three sections per family demonstrates that the classifier's high performance is concentrated in a small number of binary regions. Besides verifying the xAI methods, this also implies that the classifier primarily focuses on local patterns found in specific sections, compared to global patterns across the entire image.

The less consistent results in the reverse experiment where occluding the least important sections still degrades accuracy at higher values of k, with Grad-CAM dropping notably at k=3, suggest that the methods are less reliable at identifying genuinely unimportant regions than at identifying important ones. SHAP's relatively strong performance in the reverse experiment, maintaining 85.7% accuracy even at k=5, suggests it produces the most conservative and reliable bottom rankings of the three.

5.3 Comparison of Methods

SHAP achieves the largest accuracy drop at $k = 1$, making it the most precise method for identifying the single most important section per family. Grad-CAM achieves the largest drop at $k = 2$ and performs comparably to SHAP thereafter, while also being orders of magnitude faster due to requiring only a single backward pass. LIME consistently performs worst in the top-section occlusion experiment and performs poorly in the reverse experiment as well, suggesting it is the least reliable of the three for this task.

However, the high Spearman rank correlations between all method pairs ($\rho > 0.90$ across families) suggest that despite differences in magnitude and specific rankings, the three methods broadly agree on which sections matter. This convergence provides additional confidence that the identified sections such as `.rdata`, `.data`, and `.eh_frame` for PE binaries are genuinely important features for the classifier rather than results of any one explainability technique.

When it comes to efficiency, Grad-CAM clearly outperforms the other methods, being 50-60 times faster due to only requiring one evaluation per sample. While the other 2 methods could be sped up by reducing their evaluations per sample, the current parameters are already on the lower end compared to what is recommended for each method.

5.4 Comparison with Prior Work

This work confirms and extends the findings of the studies mentioned in sections 2. Where prior studies established that xAI heatmaps are visually informative for malware image classifiers, this paper demonstrates that these heatmaps correspond to specific, interpretable binary sections, and that their importance can be quantitatively validated through occlusion. The section-mapping procedure allows for more specific conclusions to be made when applying xAI to malware image classifiers.

5.5 Limitations

Several limitations of this study should be acknowledged.

First, the dataset consists of neutered malware samples, which preserves signatures and structure but removes functional behavior. While this is appropriate for classification purposes, it is possible that the byte patterns in some sections differ from those of functional samples.

Second, the xAI methods have only been used in one specific configuration. Analyzing different layers using Grad-CAM or using different amounts of samples for LIME and SHAP may have different results.

Third, section mapping is performed at the row level rather than the pixel level due to information loss from resizing; this provides a useful approximation but will still have information from multiple rows bleeding into one row, and will merge the attention of multiple small sections on the same row.

Finally, the findings are specific to ResNet-18 and the chosen dataset. Generalisability to other architectures or datasets cannot be assumed without further experimentation.

6 Conclusions and Future Work

This paper investigated which Explainable AI method is most effective for explaining CNN decisions on malware byteplot images, and how the methods differ in their explanations. A ResNet-18 classifier achieving 99.93% accuracy was trained on a 14-family malware dataset, and Grad-CAM, LIME, and SHAP were applied to its test-set predictions. For each sample, highlighted image regions were mapped back to named binary sections, and an occlusion experiment was conducted to validate that high-attention sections are genuinely critical to classification.

All three methods consistently identify `.rdata`, `.data`, and `.eh_frame` as the most important sections for PE binary classification, with `.rdata` ranked highest across all methods. For ELF binaries, attention is distributed more broadly across sections, reflecting the greater structural diversity of the ELF malware families in the dataset. SHAP diverges notably from Grad-CAM and LIME on ELF binaries, likely due to its finer-grained, more local attribution approach.

The occlusion experiments confirm that the identified sections are genuinely important: occluding only the top three sections per family causes accuracy to drop below 50% for all three methods, well exceeding the random baseline. In the reverse experiment, SHAP proves the most conservative and reliable method, maintaining 85.7% accuracy even when the five lowest-ranked sections are occluded.

Comparing method effectiveness, SHAP identifies the single most critical section most precisely (largest accuracy drop at $k = 1$), while Grad-CAM achieves comparable overall performance and is orders of magnitude faster due to requiring only a single backward pass. LIME performs worst in both the top- and bottom-section occlusion experiments, suggesting it is the least reliable of the three for this task. Despite these differences in performance and ranking magnitudes, Spearman rank correlations above 0.90 across all method pairs indicate broad agreement on which sections matter most, lending additional confidence to the identified features.

Taken together, these findings show that CNN-based malware image classifiers primarily focus on local patterns found in specific sections, compared to global patterns across the entire image. For analysts, Grad-CAM offers the best trade-off between effectiveness and efficiency. SHAP is preferable when identifying the single most important region matters most, or when reliable identification of unimportant regions is needed. LIME offers little advantage over the other two methods for this task.

One major concern is the finding that the classifier relies quite a lot on sections such as `.eh_frame`. Since these sections do not reflect the actual behavior of files, and adversaries can easily modify these sections without affecting the core functionality of their malware, the classifier could easily be misled. Therefore, the high accuracies of image based classifiers should not be taken at face value, and models should first be thoroughly analyzed before being used in any production environment.

Future work should examine whether these findings generalize to other CNN architectures, larger and more diverse datasets, and non-neutered malware samples. Applying these methods to a classifier that does not require resizing of the original images would also result in greatly increased accuracy when it comes to smaller sections. Lastly, testing different configurations of the used xAI, or other xAI methods, could also lead to more insight.

7 Responsible Research

7.1 Reproducibility

The methods used in this study are based entirely on publicly available libraries and a well-known model architecture. All parameters, segmentation settings, and evaluation procedures are reported in full in Section 3, and the code is published in appendix C, enabling independent reimplementation. One limitation to full reproducibility is that the dataset was provided privately by the thesis supervisor and is not publicly available. This means that exact replication of the experimental results is not possible without access to the same samples. However, the research should still be reproducible with a similar dataset, provided it contains the same components.

7.2 Risks of Misuse

One concern inherent to xAI research in cybersecurity is that publishing which features a classifier relies on can aid adversaries in evading detection. By revealing that a ResNet-18 malware classifier places significant attention on sections such as `.rdata`, `.data`, and `.eh_frame`, this paper provides relevant info to adversaries on which file sections to manipulate to avoid correct classification. The finding that `.eh_frame` receives high attention is particularly useful for an adversary, since stripping or altering it requires no modification to the malware's core functionality. This concern is explicitly acknowledged in this paper and is presented as an argument for caution before deploying such classifiers in production. On the other hand, these vulnerabilities are also important information for defenders, as it helps design more robust classifiers. Lastly, malware image based classification currently sees practically no use in the real world, so there is relatively no risk of these findings affecting malware detection systems that are currently being used.

7.3 Dataset

Since the dataset fully consisted of neutered samples, no functional malware was handled, executed, or distributed during this research. Despite being neutered, the dataset should still be reliable for this research as the samples have the same signatures. Because the samples are balanced across 14 families with 715 samples each, results are unlikely to be strongly skewed by class imbalance.

7.4 Use of Large Language Models

Large Language Models (LLMs) were used to assist in structuring this report, and finding grammatical errors. They were

also used to generate boilerplate code for sections such as visualizations. All content generated by LLMs was critically reviewed, verified, and revised to ensure correctness and appropriateness for this project

References

[1] AV-TEST Institute, "Malware statistics & trends report," 2024. Accessed: 2026-05-27.

[2] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, (New York, NY, USA), Association for Computing Machinery, 2011.

[3] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of cnn architectures (imcec)," *Computers Security*, vol. 92, p. 101748, 2020.

[4] D. Pant and R. Bista, "Image-based malware classification using deep convolutional neural network and transfer learning," pp. 1–6, 11 2021.

[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, p. 336–359, Oct. 2019.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016.

[7] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[8] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Towards an interpretable deep learning model for mobile malware detection and family identification," *Computers & Security*, vol. 105, p. 102198, 2021.

[9] M. Brosolo, V. P., and M. Conti, "Through the static: Demystifying malware visualization via explainability," *Journal of Information Security and Applications*, vol. 91, p. 104063, 2025.

[10] S. Nazim, M. Alam, S. Rizvi, J. Mustapha, S. Hussain, and M. Suud, "Advancing malware imagery classification with explainable deep learning: A state-of-the-art approach using shap, lime and grad-cam," *PLOS One*, vol. 20, 05 2025.

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[14] J. Gildenblat and contributors, "Pytorch library for cam methods." https://github.com/jacobgil/pytorch-grad-cam, 2021.

[15] M. T. C. Ribeiro and contributors, "Lime library." https://github.com/marcotcr/lime, 2020.

[16] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Computer Vision – ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), (Berlin, Heidelberg), pp. 705–718, Springer Berlin Heidelberg, 2008.

[17] S. Lundberg and contributors, "Shap library." https://github.com/shap/shap, 2026.

[18] A. C. Telea, "An image inpainting technique based on the fast marching method," *Journal of Graphics Tools*, vol. 9, pp. 23 – 34, 2004.

A Full family section attention heatmaps

A.1 PE binaries

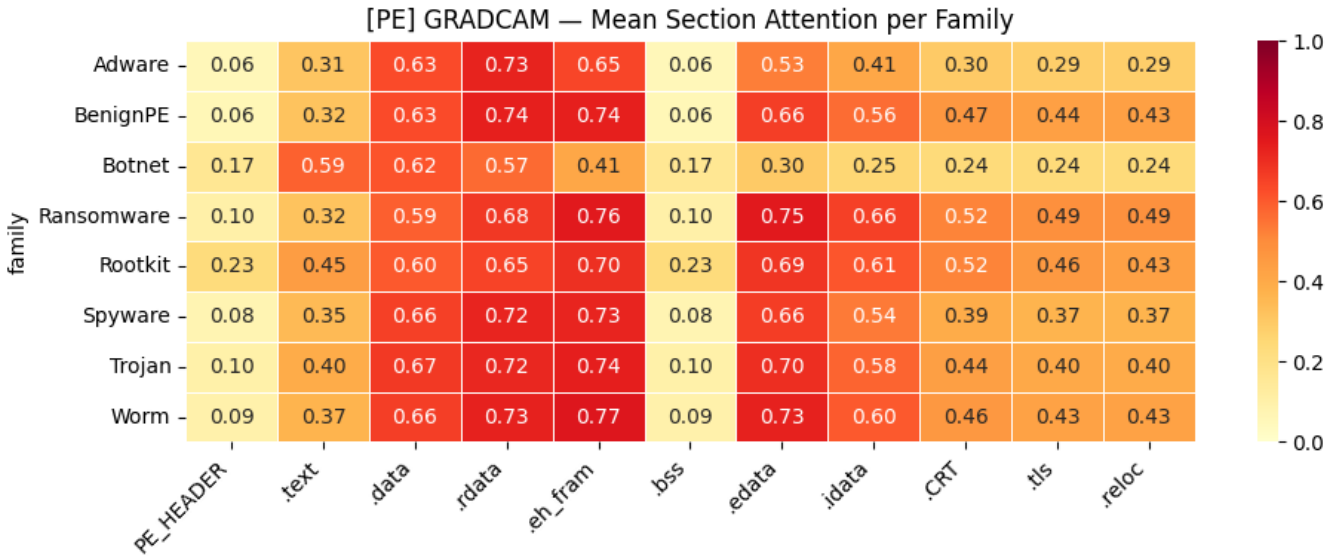

Figure 6: Grad-CAM attention per section per family for PE files

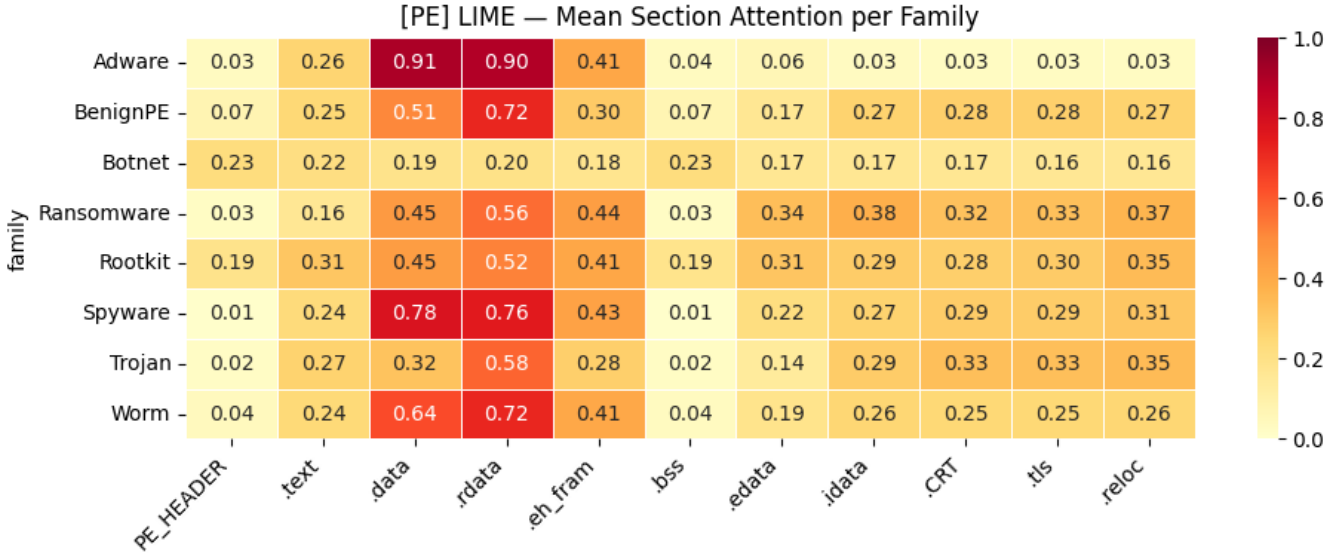

Figure 7: LIME attention per section per family for PE files

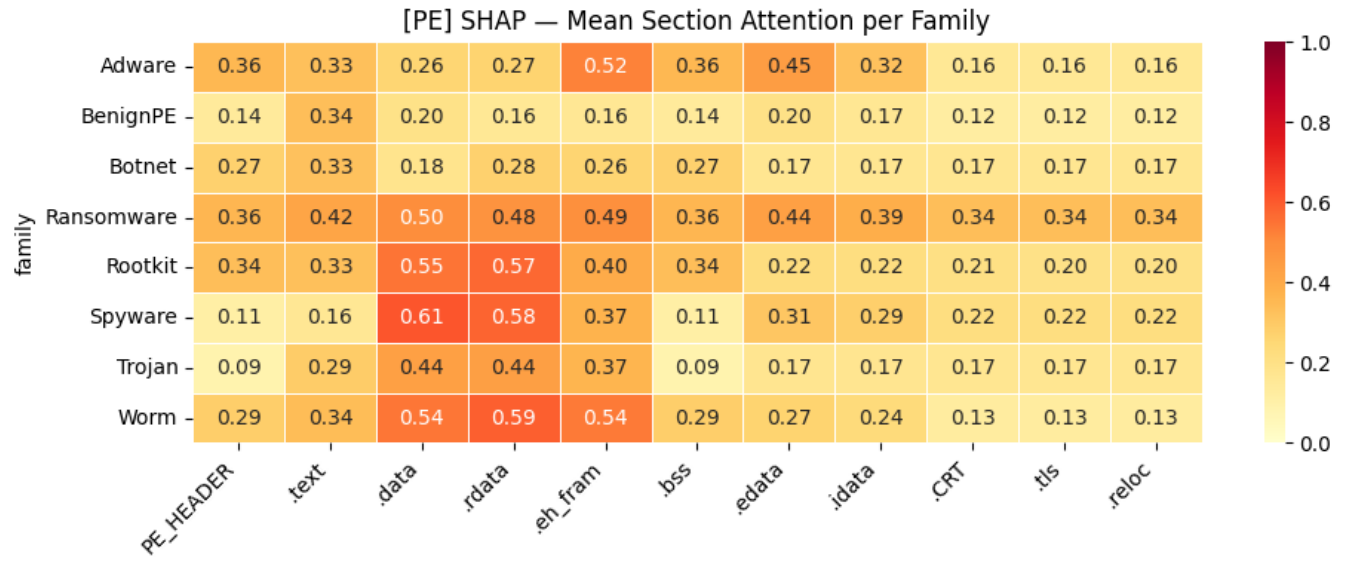

Figure 8: SHAP attention per section per family for PE files

A.2 ELF binaries

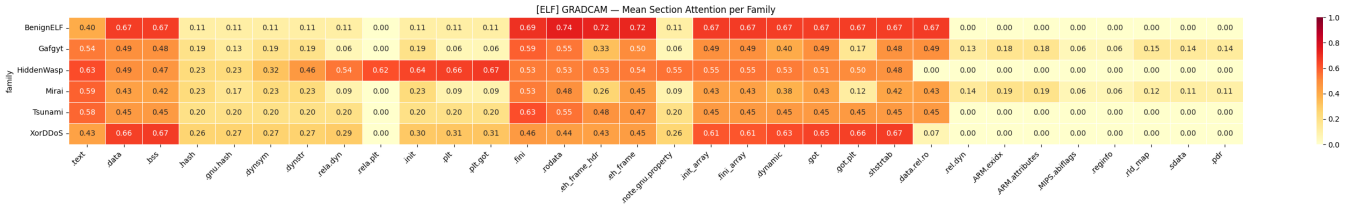

Figure 9: Grad-CAM attention per section per family for ELF files

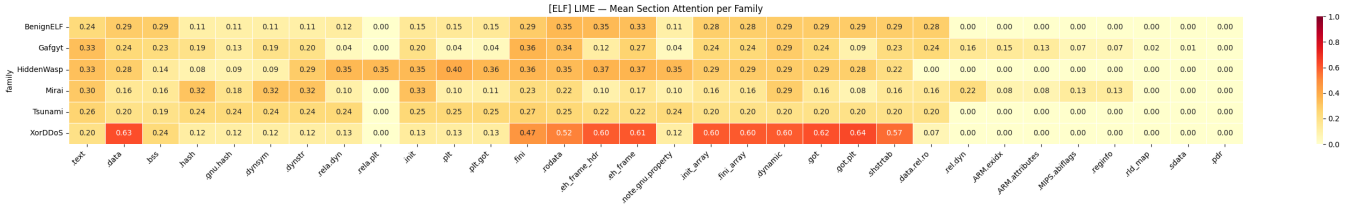

Figure 10: LIME attention per section per family for ELF files

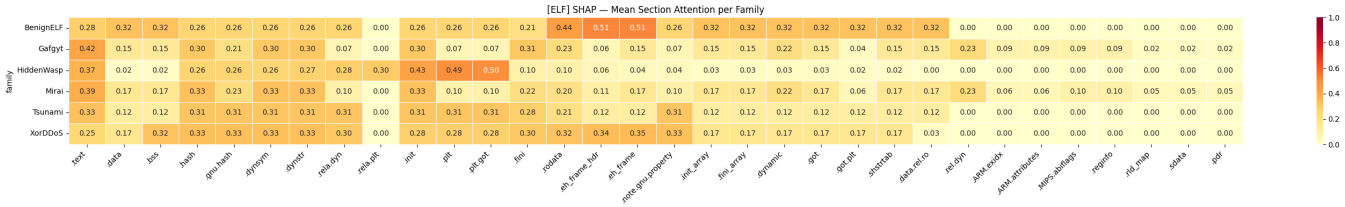

Figure 11: SHAP attention per section per family for ELF files

B Noise Occlusion Experiment Results

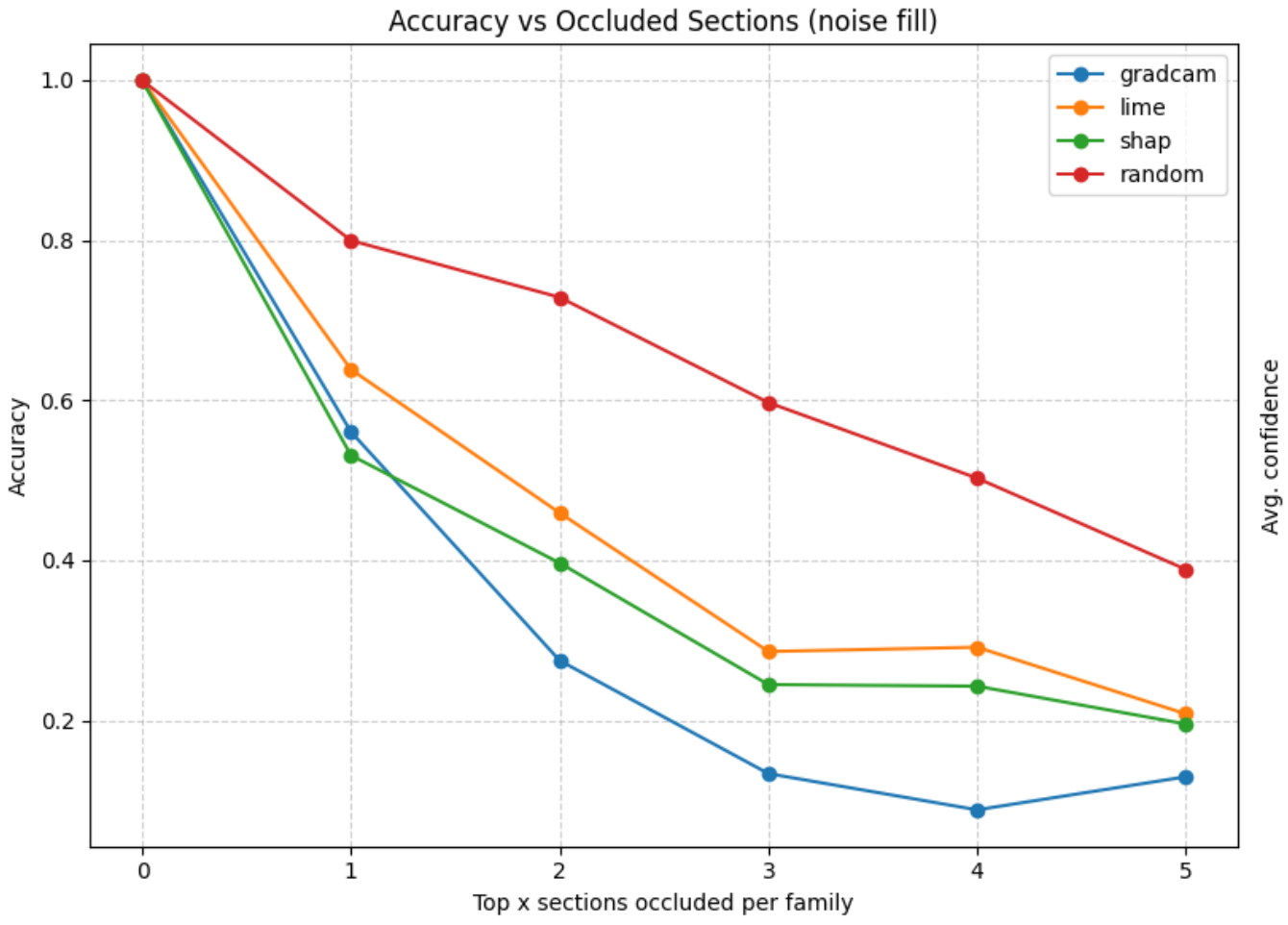

Figure 12: Accuracy drop after occluding the top k sections for each method.

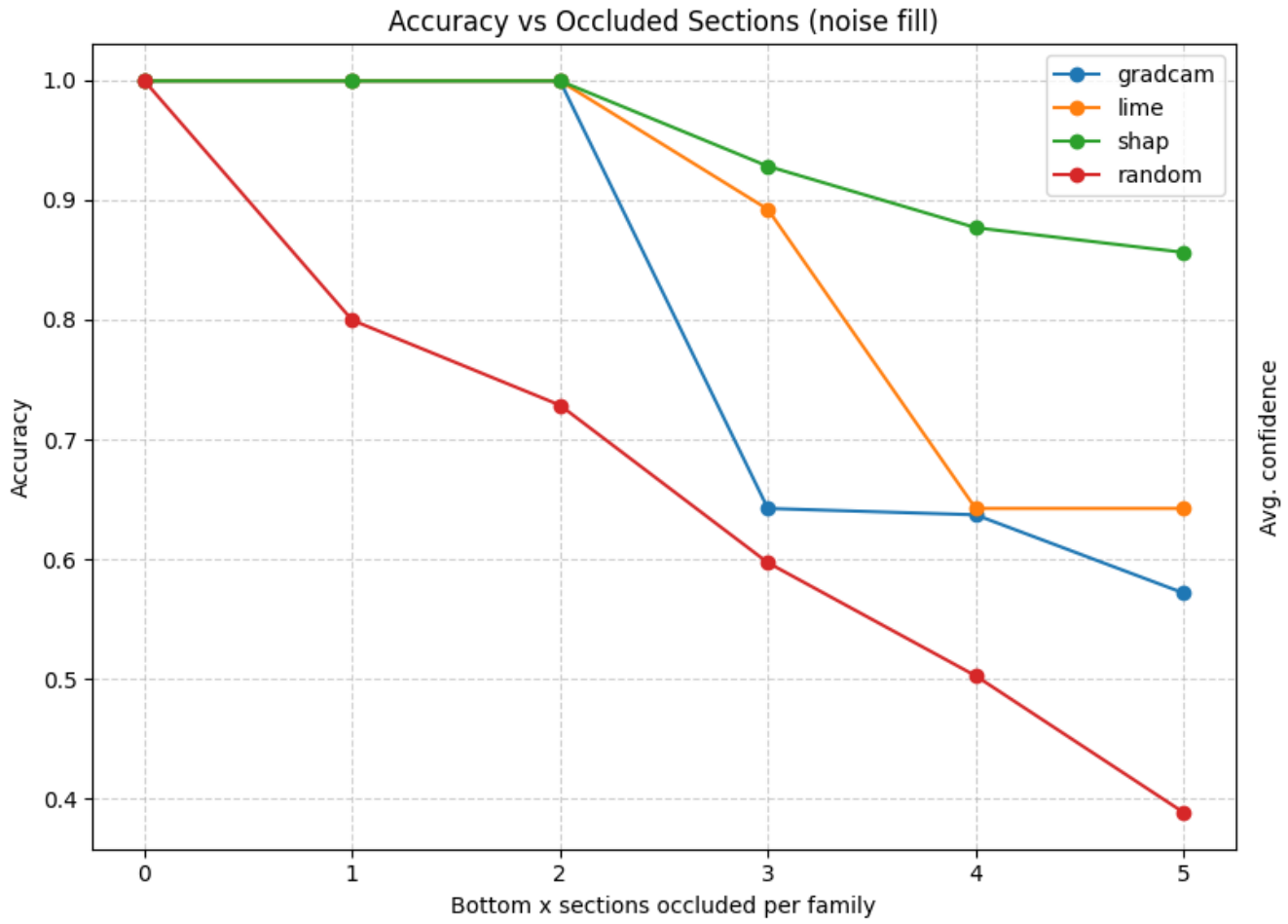

Figure 13: Accuracy drop after occluding the bottom k sections for each method.

C Source Code

The code for all the experiments is available at https://github.com/BennbuildVGC/xai-malimg