Malware Through the Lens of Computer Vision

How Binary-to-Image Encodings Influence CNN-Based Malware Family Classification

Martim de Assis Lopes Cardeira

Supervisor(s): Tom J. 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: Martim de Assis Lopes Cardeira
Final project course: CSE3000 Research Project
Thesis committee: Tom J. Viering, Akash Amalan, Georgios Smaragdakis

Abstract

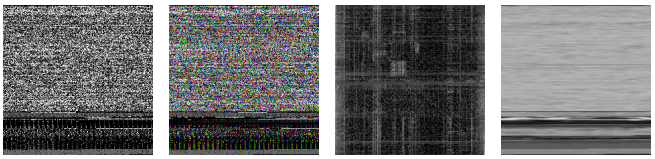
Image-based malware classification through binary-to-image encodings has become a popular, quick-to-run and disassembly-free alternative to signature-based methods. Near-perfect accuracies are often reported, alongside weakly substantiated claims of resilience to malware obfuscation by adversaries. We present the first controlled, head-to-head comparison of four such encodings - grayscale and RGB byteplots, Markov bigram plots, and sliding-window Shannon entropy - under a fixed ResNet-18 classifier and a balanced dataset spanning seven packer conditions. To test whether high accuracy reflects genuine family structure or exploitable shortcuts, we pair the comparison with a diagnostic framework combining random-forest feature baselines, TLSH similarity analysis, HiResCAM and occlusion-based explainability methods. No single diagnostic suffices; only their combination separates shortcut-driven scores from genuine learning. We find that encoding performance cannot be judged by accuracy alone; models may fit to shortcuts such as file-size and near-duplicate samples, rather than learning malware structure. Entropy is found to be the dominant and most obfuscation-resilient encoding, while exhibiting the most evidence of genuine family-discriminative learning.

1 Introduction

The accurate and efficient classification of malware has proven to be a continuous struggle in the area of cybersecurity. Traditional signature-based detectors are faced with a steadily growing stream of new malware variants year on year, underlining the need for alternatives that generalize well to unseen and obfuscated variants, without manual feature extraction or costly runtimes.

Nataraj et al. proposed one such alternative in 2011, showing that CNNs trained on grayscale image representations of malware could classify the family of unseen variants with up to 99% accuracy [1], attributing the result to the visual similarity of samples within a family. The approach requires neither disassembly nor execution, and its authors claimed resilience to obfuscation [1]. It prompted a line of research pursuing higher accuracies through more powerful classifiers and, of central interest to this paper, through alternative binary-to-image encodings which include: RGB byteplots [2], Markov bigram plots [3], and sliding-window Shannon entropy plots [4]. A recurring theme across these encodings is the strongly stated but weakly substantiated claim that they resist malware obfuscation.

We study these four encodings as they represent distinct paradigms rather than variations on a single idea. The grayscale byteplot and its denser RGB extension preserve the binary's spatial layout and byte values. The Markov plot captures byte-transition statistics to build a picture of program instruction semantics and control flow. Lastly, the sliding-window Shannon entropy captures the randomness over the regions of the binary, highlighting encrypted or compressed sections. Each encoding assumes the family-discriminative features live in a different domain, yet all reduce to a 2D image (Figure 1).



(a) Grayscale (b) RGB (c) Markov (d) Entropy

Figure 1: Four image encodings of the same **Mirai** malware sample.

The near-perfect results of these methods have raised concerns about what the underlying models are in fact learning [5; 6; 7]. A growing body of work suggests headline accuracies may be grossly inflated due to unbalanced datasets and malware samples that are near-identical at the byte level [8; 9].

Packing, referring to code obfuscation techniques, has been shown to degrade image-based classifiers sharply [7]. Decisive regions of a malware image are often found to be fixed structural elements such as PE headers rather than malicious code patterns [7], exhibiting *shortcut learning* [10] where signals that merely correlate with the label in the dataset are exploited in place of meaningful learning.

Recent work extensively explores malware classification across individual binary-to-image encodings, each with a varying degree of focus on performance under obfuscation [11]. Crucially, however, surveys find there is very little work done on controlled, head-to-head comparisons between encodings, and individual results cannot be compared as authors use different classifiers and datasets [11; 12]. Lastly, most explainability papers focus on grayscale byteplots as they are the default encoding in this field, with explainability of other image-encodings being absent [11]. This paper addresses the aforementioned gaps by asking:

> *Across four binary-to-image encodings (grayscale byteplot, RGB byteplot, Markov bigram plot, and sliding-window Shannon entropy), how well do CNNs classify malware under obfuscation, and how much of that reflects genuine family-discriminative structure?*

We address this through the following sub-questions:

- **RQ1.** How do accuracy and macro-F1 compare across the four encodings and the selected packers?
- **RQ2.** For each encoding, is high accuracy attributable to genuine family-discriminative structure or to shortcut signals, e.g. file-size, near-duplicate variant leakage?
- **RQ3.** Which regions of the encoded image does the model rely on; do they correspond to specific code sections or to headings and padding?

The contributions of this paper are as follows:

1. A controlled comparison of Grayscale/RGB byteplots, Markov bigram plots and sliding-window Shannon en-

tropy encodings on a fixed ResNet-18 architecture and a balanced dataset.
2. A per-packer evaluation across 7 conditions (an unpacked baseline and 6 diverse packers), isolating how each encoding behaves under different obfuscation strategies.
3. A diagnostic framework combining random-forest baselines, TLSH leakage analysis, and HiResCAM/ablation that work in conjunction to detect shortcut exploitation.

2 Related Work

In this section, we situate our work within the broader context of existing research and discuss key developments relevant to our study, as well as highlight knowledge gaps that we aim to address. We organize prior work into two strands: the binary-to-image encodings whose comparison is the subject of this paper (Section 2.1), and the body of work questioning what such models actually learn, which motivates our diagnostic framework (Section 2.2).

2.1 Binary-to-Image Encodings for Malware Classification

Grayscale byteplots, introduced by Nataraj et al. in 2011 [1], read a binary as a sequence of bytes representing grayscale pixel intensities, yielding a 2D image that captures spatial byte information. The intuition is that visual textures are strikingly similar within a malware family yet distinct across families, an effect attributed to the heavy code reuse among variants of the same family. The approach requires neither disassembly nor execution, and on a benchmark of 9,458 samples it achieved roughly 97-99% classification accuracy. Going beyond basic family classification, the paper made a bold yet mostly unsubstantiated claim about resilience to obfuscation by packers. The unbalanced MalImg dataset used in the evaluation limits the findings.

A natural extension of Grayscale was proposed by Vasan et al. [2]. Their IMCFN model takes advantage of the color channels to read three subsequent bytes of a binary into a single color pixel, resulting in a denser encoding with a reported 1.64% increase in accuracy over the Nataraj model on the same dataset. Similar, yet weakly substantiated, claims were made about resilience against obfuscation methods [2].

Branching away from byteplot methods, the modeling of malware binaries as Markov State Transition Probability Matrices (STPM) has been explored [13; 3]. The key intuition behind STPMs is that they capture byte-transition probabilities, which contain information about instruction sequences and control flow of the program. With a heavy focus on capturing the actual semantics of the binary (i.e. what the program is doing), the paper reported extremely high accuracies of 99.8% and a claimed resilience to obfuscation methods such as UPX [3], although it was admitted that due to the choice of datasets generalization might be overestimated [3].

A different approach relies on Shannon entropy. Lyda and Hamrock demonstrated how computing the Shannon entropy of different sections of a binary could spot potentially malicious packing or encryption attempts [14]. Building on that intuition, Han et al. experimented with computing the Shannon entropy over a sliding window across the binary to achieve a 1-D signal [15]. Gibert et al. claim strong *packer detection*, although they show actual malware classification degrades noticeably under compression/encryption [16]. Darton represented sliding window entropy as a square image, the encoding investigated in this paper, and found it outperformed Grayscale, although no direct claims were made about obfuscation [4].

2.2 Evaluation Critiques and Shortcut Learning

Alongside the pursuit of higher accuracies, a growing body of work questions whether the reported numbers reflect genuine learning. Several studies observe that benchmark datasets such as MalImg are dominated by families whose samples are near-identical at the byte level; under cross-validation splits, models can succeed by recognizing near identical samples in other folds, rather than by generalizing to unseen malware [8; 9]. Packing has been shown to degrade image-based classifiers sharply, and several studies argue that such models frequently learn to identify the packer rather than the malware it conceals [7; 8]. Works that claim packing does not significantly degrade performance typically examine packers that leave significant structures intact, such as headers and bytegrams [17]. Explainability analyses reinforce this, finding that the decisive regions of a malware image are often fixed structural elements such as PE headers rather than actual malicious code [7]. These observations are instances of shortcut learning, in which a model attains high accuracy by exploiting a signal that correlates with the label but does not reflect the underlying semantics [10]. Our work builds directly on this strand, extending it from observation to a controlled, multi-encoding study on a shared dataset.

3 Methodology

This section details the experimental setup. To isolate the effect of the encodings on family classification, each encoding is trained and evaluated under a common CNN architecture and parameters for each packer individually. We describe the dataset (Section 3.1), the four image encodings (Section 3.2), the fixed ResNet-18 classifier and training procedure (Section 3.3), and the experimental design (Section 3.4). Because a high score may reflect a shortcut rather than genuine family structure, Section 3.5 introduces a diagnostic framework that explores and highlights potential shortcuts for each (encoding × packer) pair.

3.1 Dataset

The dataset consists of 10,010 neutered, unpacked samples of both benign and malware binaries spanning 14 different families. Eight families are in the Windows PE format: **Trojan, Ransomware, Worm, Spyware, Adware, Rootkit, Botnet, and *BenignPE***. Six families are in the ELF format: **Mirai, Gafgyt, HiddenWasp, XorDDoS, Tsunami, and *BenignELF***. All families are equally represented in this dataset.

In order to study robustness under obfuscation, a subset of samples are also provided with their packed counterparts, totaling 11 different packing methods. *Mangle* and *MPRESS* do not support ELF binaries, hence they only represent the 8 PE

families. To reduce redundancy in studying similar packing methods, 6 of the 11 different packers are selected, focusing on the variants with the most structural disruption while maintaining diversity. A breakdown of the selected packers and their sample counts is given in Table 1.

The packers represent three distinct obfuscation strategies. *MPRESS, UPX, zlib* and *ZIP* rely on **compression**. The former two create a runnable binary with a code stub that decompresses the actual malware at runtime, whereas the latter two create an archive file. *XOR* works on the principle of **encryption**, applying byte-level obfuscation by XORing each byte with a key that is randomly generated through a keystream. Lastly, *Mangle* relies on **code-level substitution**, replacing common Indicators of Compromise (IoCs) such as file hashes or domains with random characters, as well as cloning code-signing certificates of legitimate PE files.

Table 1: Dataset overview per packer.

Packer	Families	Samples	Type
None	14	10,010	Baseline (unpacked)
MPRESS	8	520	Compression
UPX	14	996	Compression
zlib	14	997	Compression
ZIP	14	997	Compression
XOR (rolling key)	14	997	Encryption
Mangle	8	520	Substitution
Total		**15,037**	

See Section 7.1 for availability.

3.2 Image Encodings

A total of four encoding methods are evaluated, each with their own characteristics.

Grayscale Byteplot [1]: Each byte in the binary is linearly mapped to an integer pixel intensity value in [0, 255] (where *FF* maps to 255 and *00* maps to 0), resulting in a grayscale image. The width of the image is determined by the width table in [1], as given by Table 11 in Appendix B, assigning a fixed width based on the file-size with height being variable. Finally, the resulting image is resized to 224×224 through bilinear interpolation, of which the implications and motivation are further discussed in Section 3.3. This encoding preserves the spatial layout of different sections of the binaries, and serves as the baseline method.

RGB Byteplot [2]: Similar to *grayscale byteplot*, with the difference being that every three consecutive bytes form one RGB pixel, with R=b_0, G=b_1, B=b_2 [2]. The mapping logic from byte values to pixel (in this case channel) intensities remains identical. The same width table from [1] determines layout and the resizing procedure is identical. This encoding achieves a $3\times$ denser representation of the binary compared to grayscale byteplots while preserving the same spatial structure, taking full advantage of the RGB channels.

Markov Bigram Plot [3]: A 256×256 state transition probability matrix, P, is constructed by counting the frequencies of all consecutive byte pairs (b_i,b_{i+1}) in the binary, such that the entry $P[i][j]$ is the number of times byte i is followed by byte j. P is then row normalized to 1 to transform the frequencies into transition probabilities. A power-law gamma mapping $M' = 255 \cdot P^{\gamma}$ is applied element-wise prior to uint8 conversion to prevent low-probability transitions from being clamped to zero. Finally the image is resized via the aforementioned method. This encoding captures byte-pair statistics and is particularly sensitive to instruction-level patterns in the binary. Nie and Zhu found the choice of γ to be dataset dependent [3]; an optimal γ value of 0.5 was determined empirically, as detailed in Appendix A.

Shannon Entropy [4]: Each binary is encoded as a one-dimensional entropy signal. A fixed-length window slides across the binary; at each position, a histogram of the window's byte-values is computed (denoting the probability distribution of the 256 possible byte-values in the window). From the histogram, the Shannon entropy $H = -\sum_{i=0}^{255} p_i \log_2 p_i$ is calculated. As the range of H is [0, 8] bits, we linearly scale the result to [0, 255] to be interpreted as grayscale pixel intensity values. The 1-D entropy signal is finally reshaped into a square image via the same width table in [1], and resized for the CNN similar to the other encodings. A window size of 256 bytes and stride of 16 bytes were chosen empirically. Appendix A details the full analysis and motivation of the parameters.[1]

3.3 Model Architecture and Training

A ResNet-18 CNN is used for all experiments. The model is trained from scratch without any pretrained weights. All images are resized to 224×224 via bilinear interpolation. Bilinear interpolation is used as opposed to nearest neighbor in order to avoid adverse effects of aliasing, which could heavily impact highly noisy images - either as a result of the encoding or intense obfuscation. Models are trained using the Adam optimizer with a learning rate of 1×10^{-4} and CrossEntropyLoss for 30 epochs with a batch size of 64. Training is performed on an NVIDIA A40 48GB GPU.

3.4 Experimental Design

Each model is trained independently per (encoding × packer) pair, yielding 28 combinations. Model architecture and hyperparameters are held constant across all conditions to ensure a fair comparison. Grayscale is run as a control variable, serving as a baseline comparison to the other three encodings. Each model is evaluated only on the packer it was trained on. Each packer contains only unique malware samples, therefore splitting the sets by SHA is implicitly handled.

To obtain statistically reliable estimates, stratified 5-fold cross-validation is used with a fixed seed, ensuring balanced family distribution across folds. For PE-only packers (Mangle, MPRESS), we stratify over the 8 available PE families. Classification performance is measured using accuracy and macro-F1 score, and reported as mean ± SD across folds.

3.5 Diagnostic Framework

A high score by itself does not guarantee that a model has truly learned family-discriminative features as opposed to some shortcut, therefore the (encoding $\times$ packer) results are examined with several tools.

[1]The encoding shorthands are hereby defined as: Grayscale, RGB, Markov, Entropy

Random Forest Classifier

A simple Random Forest (RF) classifier is trained on the following feature set: log file-size in bytes (logarithmic due to the large range of magnitudes in the dataset), the global Shannon entropy of the binary, and the mean and standard deviation of the Shannon entropies of 4000 linearly spaced 256 byte windows. File-size is a trivial shortcut when families have tight and distinct size distributions, whereas global entropy, and entropy distribution summaries are subtler shortcuts that can be inferred from the image, for instance by the overall noise, or distribution of noise in Grayscale. Because the entropy features overlap with what Entropy itself captures (although only scalar quantities), we additionally train two RFs on size only and entropy only, isolating each shortcut's individual contribution. All RFs use the same 5-fold cross-validation protocol established in Section 3.4. High RF accuracies indicate viable shortcuts the CNNs may exploit, prompting further investigation via complementary methods.

TLSH Similarity Analysis

Following Gibert et al. [8], the TLSH locality-sensitive hashing algorithm is used to determine near-duplicate binaries by calculating pairwise hash distances. We adopt the conservative and permissive distance thresholds of 30 and 70, as supported by the results of the original TLSH paper [18].

For each packer, we replicate the 5-fold cross-validation split from Section 3.4 and calculate *leakage* - the rate of binaries with at least one near-duplicate in a different fold. A high rate across different folds is akin to test snooping [9], which can lead to memorization of samples rather than learning. This inflates accuracies and overstates the model's ability to generalize to unseen variants.

Attention Analysis

To understand how each model is distributing its attention, the procedure proposed by Brosolo et al. [5] is adapted. First, for each pair we compute cumulative, per-family heatmaps using HiResCAM [5]. HiResCAM is preferred over Grad-CAM, as Draelos and Carin proved it to be more faithful in its explanations [19].

Occlusion is then applied, although we investigate the opposite of [5]. Their paper occludes non-important regions of images according to the cumulative heatmaps in an attempt to increase classification performance. Conversely, we opt for *band ablation*, in which we progressively occlude the top and bottom N bands with zeroes. The top and bottom occlusions are done separately, and at inference time. This is done to observe how performance degrades once the images have been stripped of header and padding information respectively, potentially revealing shortcuts relating to those regions.

4 Results

This section presents the results for the Classification Performance (**RQ1**), Random Forest baseline and TLSH Similarity Analysis (**RQ2**), and the Attention Analysis (**RQ3**).

4.1 Classification Performance

Figure 2 depicts the test accuracies over all encoding × packer pairs, answering **RQ1**. Macro-F1 closely tracks accuracy across all pairs (Tables 12, 13), diverging only modestly on the lowest-scoring cells, as expected on a class-balanced dataset; we therefore discuss accuracy in the main text and defer F1 to Appendix B.

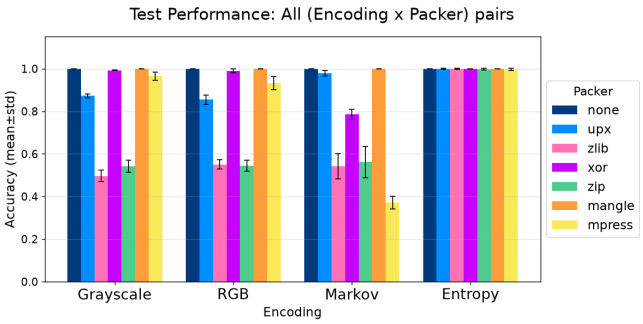


Figure 2: Per-packer accuracy across the four encodings (5-fold CV, error bars show SD).

Four observations follow directly from these results.

Perfect performance on none and Mangle; high performance on UPX across all encodings. On the unpacked baseline and Mangle, every encoding reaches 1.000 ± 0.000; UPX is nearly as high (0.855–0.999).

RGB mirrors Grayscale. For all packers except zlib and MPRESS, the observed difference between both is ≤ 0.017, and they sit within the margin of error of each other. RGB beats Grayscale by 0.055 on zlib, falling outside the margin of error; conversely Grayscale beats RGB on MPRESS by 0.032, however just within the margin. As the two display a similar level of performance across all packers, RGB is dropped in further analysis for tractability.

Entropy is robust under all packers. Entropy reports an accuracy ≥ 0.996 across all packers, which aligns with its unpacked performance. Performance does not degrade under zlib and ZIP unlike the other encodings, demonstrating particular resilience under compression.

Markov fails where the others do not. Markov is the only encoding to degrade significantly on XOR (0.786 vs. $\geq$ 0.990) and produces the worst result in the study on MPRESS (0.371) by a significant margin.

4.2 Random Forest Baseline

Table 2 presents the results of the RF baselines described in Section 3.5. The baselines are compared to the overall best CNN encoding models with and without the inclusion of the entropy models, to further determine the importance of learned entropy features on performance.

Three observations follow directly from these results.

File-size provides near-perfect performance. With the exception of MPRESS, every packer attains an accuracy $\geq$ 0.940 solely on the most trivial of features, indicating an extremely vulnerable shortcut available to the CNNs. **MPRESS is the big exception**; manual inspection of the binaries shows that the Botnet family sits firmly in the (95KB–96KB) range, meanwhile all other families are within the (25KB–29KB)

Table 2: Random-forest baselines vs. CNNs, per packer. The baseline is trained on three scalar feature sets: logarithmic file-size, global/windowed entropy statistics, and both combined.

Packer	RF (size)	RF (entropy)	RF (size+ent.)	CNN (best)	CNN (excl ent.)
none	1.000	0.989	0.999	1.000	1.000
UPX	0.983	0.892	0.968	0.999	0.979
zlib	0.950	0.700	0.924	0.999	0.551
XOR	0.999	0.829	0.975	1.000	0.993
ZIP	0.940	0.669	0.938	0.998	0.562
Mangle	1.000	0.938	0.985	1.000	1.000
MPRESS	**0.525**	0.729	0.865	0.996	0.965

range. This is a much tighter file-size distribution than other packers. Inspection of the confusion matrices reveals that *no encoding misclassified Botnet across any fold.*

File-size outperforms scalar entropy statistics. In all but the MPRESS packer, the file-size RF baseline outperforms the entropy equivalent. This highlights file-size as being simultaneously more trivial, and more discriminative than entropy measures.

The MPRESS exception. MPRESS breaks the pattern, as the file-size and entropy baselines reach only 0.525 and 0.729 respectively. The combined baseline reaches an accuracy of 0.865 which is surpassed by all encodings except Markov.

4.3 TLSH Similarity Analysis

Table 3 presents the results of the TLSH similarity calculated across 5-fold cross validation splits used in CNN training as described in Section 3.5.

Table 3: Near-duplicate leakage per packer: fraction of samples with a near-duplicate in a *different* fold, at two TLSH thresholds.

Threshold	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
TLSH ≤ 30	100.0	56.9	0.0	54.5	0.0	100.0	0.0
TLSH ≤ 70	100.0	69.5	4.8	60.6	4.3	100.0	0.0

The results partition the packers into three distinct classes.

Saturated leakage: none and Mangle. Every unpacked and Mangle-packed sample has a near-duplicate in a different fold at both thresholds. This correlates strongly with the accuracies achieved by all four encodings on these packers.

Partial leakage: UPX and XOR. Roughly 55–70% of samples in the UPX and XOR datasets have near-duplicates across fold boundaries. The relatively high accuracy on these packers could be attributable to data leakage.

Clean splits: zlib, ZIP, and MPRESS. These methods are sufficiently transformative to break near-duplicate similarity at threshold 30, with leakage below 5% even at a more permissive threshold of 70. MPRESS produces zero near-duplicates at either threshold.

4.4 Attention Analysis

This subsection reports the attention and occlusion analyses results following the methodology outlined in Section 3.5.

HiResCAM Heatmaps

Cumulative class-level HiResCAM heatmaps were generated for every (encoding × packer) model, for each individual family. Selected examples are shown in this section for illustrative purposes and brevity. All 21 grids, totaling 258 individual heatmaps, are available in Appendix C.

Two noticeable patterns follow from these results.

Entropy produces the most family-distinct heatmaps. Compared to Grayscale and Markov, the heatmaps produced by Entropy vary significantly more between families across almost all packers. This variation of attention suggests that Entropy is learning family-discriminative structures and relying less on the broader image. Figure 3 exemplifies how Entropy heatmaps have a higher disparity on unpacked samples.

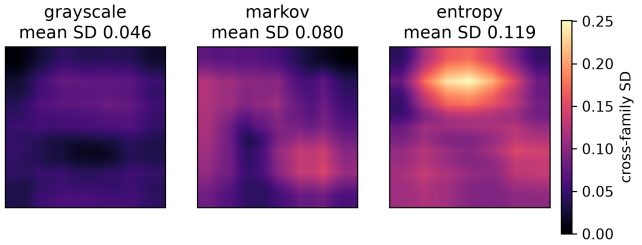


Figure 3: Per-pixel standard deviation between the different cumulative family heatmaps, per encoding on the unpacked baseline.

Grayscale distinctly focuses on code stubs in UPX and MPRESS. Attention is otherwise broad in other packers. Both of these packers rely on the execution of a stub at runtime to decompress the actual program data into memory. This stub is added to the bottom of the binary in both packers. From the heatmaps, it is evidenced that Grayscale has a strong tendency to look at the bottom region for these packers, as opposed to the high-entropy compressed segment. Figure 4 exemplifies this.

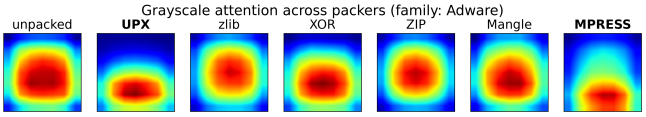


Figure 4: Cumulative Grayscale heatmaps across all packers for the Adware family.

Band Ablation

Tables 4 and 5 present the accuracies for Grayscale and Entropy respectively, after masking N top or bottom rows of pixels with zeroes during inference using the method outlined in Section 3.5. Since Markov does not preserve the layout of the binary, band occlusion is a redundant method when looking for header/padding dependence. Nonetheless, the results for Markov are available in Appendix B for completeness, exhibiting no significant degradation across all combinations.

Grayscale sees very minor degradation as a result of band ablation. Excluding MPRESS, ablation of up to 32 image

Table 4: Grayscale band-ablation accuracy. Base = unoccluded baseline; T/B = occlusion from top/bottom by band width (px).

Packer		Base	1	2	4	8	16	32
none, XOR, Mangle: baseline 1.00, 1.00 *at all bands (omitted)*								
UPX	T	.98	.98	.98	.98	.98	.97	.97
	B		.97	.98	.97	.98	.98	.98
zlib	T	.89	.89	.89	.88	.88	.87	.85
	B		.89	.89	.89	.89	.88	.88
ZIP	T	.89	.89	.89	.88	.90	.89	.86
	B		.89	.89	.89	.88	.88	.88
MPRESS	T	1.00	1.00	1.00	1.00	1.00	1.00	.99
	B		1.00	1.00	1.00	.99	.95	.85

rows degrades accuracy by at most 0.04. MPRESS only sees moderate degradation (0.15) when occluding the bottom 32 rows; this corresponds to the code stub section of the packed binary.

Table 5: Entropy band-ablation accuracy. Base = unoccluded baseline; T/B = occlusion from top/bottom by band width (px). Results equal to random guessing are highlighted in bold.

Packer		Base	1	2	4	8	16	32
none, XOR: baseline 1.00, 1.00 *at all bands (omitted)*								
UPX	T	1.00	1.00	1.00	.97	.70	.44	.63
	B		.99	.99	.99	1.00	1.00	.98
zlib	T	1.00	.92	.47	**.07**	**.07**	.14	**.07**
	B		.98	.90	.56	.25	**.07**	**.07**
ZIP	T	1.00	.52	.41	**.07**	**.07**	**.07**	**.07**
	B		1.00	.97	.89	.43	.19	.13
Mangle	T	1.00	1.00	1.00	1.00	1.00	1.00	.94
	B		1.00	1.00	1.00	1.00	1.00	1.00
MPRESS	T	1.00	1.00	1.00	1.00	.73	.23	.40
	B		.97	.88	.76	.67	.23	**.14**

Ablation results for the Entropy are far more varied than Grayscale. The packers separate into three groups.

ZIP and zlib collapse to random guessing when occluding from either end. ZIP and zlib collapse from both the top and bottom, but degrade faster on top-band ablation. Occluding a single top row drops ZIP from 1.00 to 0.52, and just 4 top rows reduce both packers to $\approx 1/14$ random guessing odds. Bottom occlusion takes 16 rows to fully collapse zlib's accuracy, while ZIP does not fully collapse (0.13 after occluding 32 rows).

UPX and MPRESS degrade inconsistently. MPRESS does not degrade on the top 1–4 rows; only once a substantial band is removed (0.73 at 8px, 0.23 at 16px) do we observe a sharp degradation, with the top and bottom borders behaving comparably. Notably, occlusion of the top 32 rows actually increases accuracy on MPRESS when compared to only 16 rows (0.40 vs 0.23). Similarly, occluding a further 16 rows from the top to a total of 32 rows increases accuracy on UPX by 0.19.

Unpacked and XOR are immune to ablation. Mangle is marginally affected. All methods retain 1.00 accuracy under removal of either border at every band width, with the exception of Mangle which degrades to 0.94 after occluding the first 32 image rows.

5 Discussion

The central contribution of this paper is providing a head-to-head performance comparison of the four different binary-to-image encodings (**RQ1**). This is answered in Section 5.1 where the strengths and weaknesses of the binary-to-image encodings are evaluated with regard to performance. Results from the framework are addressed in Section 5.2 to explain the performance disparity between encodings (**RQ2** & **RQ3**). The limitations of the study are reviewed in Section 5.3.

5.1 Encoding Performance Comparison

On both the unpacked baseline and Mangle, all four encodings classify perfectly. The major finding from our study is that **Entropy significantly outperforms and dominates all other encodings** with near-perfect accuracies (≥ 0.996), demonstrating resilience against all forms of obfuscation. Its outperformance of Grayscale is supported by [4], though their work lacks explainability. The robustness of Entropy under compression however disagrees with claims from [16].

The Grayscale and RGB encodings perform almost identically and can be discussed as a single byteplot encoding. The byteplots resist packers with low structural disruption (Mangle, XOR[2]), but degrade with increasing disruption: moderately under UPX (0.128–0.145) and sharply under zlib and ZIP (0.449–0.504). This reveals the *lack of resilience to increasingly disruptive obfuscation*. These results disprove the broad, unsubstantiated robustness claims made by [1; 2], which were made against a narrower range of packers than we test here.

Markov is the overall weakest encoding by a moderate margin. It performs within the margin of error of Grayscale, with the exception of XOR and MPRESS, on which it greatly underperforms by 0.207 and 0.594 respectively; however, it moderately outperforms on UPX by 0.107. No prior work addresses the discrepancy in robustness for Markov compared to other encodings; claims of Markov's superior *unpacked* performance against Grayscale made by [3; 13] also cannot be verified due to perfect performance of both encodings on the unpacked baseline.

5.2 Explainability of Results

Shortcuts

A high accuracy does not necessarily reflect that family structures are being learned. We examined whether the (encoding × packer) results could instead be driven by shortcuts. As highlighted in Section 4.2, file-size was found to be the most discriminative shortcut - this revealed that *in our dataset, malware families occupy highly distinct size distributions*. We cannot directly determine whether the models are exploiting this shortcut, but a few results bound how far file-size alone can account for their performance.

First, the **Botnet** family under MPRESS presents a size outlier, and every model achieves perfect precision and recall on it. In isolation, this result is consistent with file-size

[2]Read excerpt in Section 5.3 about XOR

learning. Yet, Markov **cannot exploit file-size**: its image dimensions are file-size invariant. Thus, Markov's perfect Botnet performance cannot be a size artifact, so genuine learning is evidenced. In fact, **MPRESS disproves total file-size reliance**. It compresses the remaining families into a very narrow size band, collapsing the size-only RF baseline to 0.525, yet the byteplots still reach 0.933 and Entropy reaches 0.996. Hence, the encodings must be learning features beyond size. Additionally, the entropy-only RF baseline is weaker overall than file-size, and every encoding beats it on at least three packers. This rules out total reliance on global entropy as well.

Our TLSH analysis accounts for the remaining "easy" cases. Saturated near-duplicate leakage appears in precisely the two sets that every encoding classifies perfectly (unpacked, Mangle); such similarity can lead to memorization of patterns that then fail on out-of-distribution variants [10], which in this context would be structurally dissimilar same-family samples. This suggests those headline scores may not reflect generalization of the models. MPRESS is again the counterpoint, exhibiting zero near-duplicate leakage yet ≥ 0.933 accuracy on the byteplots.

No single shortcut is sufficient to explain the observed accuracies, however we cannot rule out partial dependence on any of them. We therefore cannot answer **RQ2** definitively, but the evidence consistently points in one direction: *the models use shortcuts as an aid that complements genuinely learned family features*.

Attention Analysis

HiResCAM was deployed to visualize which regions the models were looking at when making their decisions (**RQ3**).

For **Markov**, the maps are uninformative as no distinct regions stand out. This is a shortcoming of the encoding rather than of HiResCAM. Markov discards the spatial layout of the bytes and encodes only their transition probabilities. Thus, its image structure roughly follows the (highly non-uniform) byte distribution of program binaries, as evidenced by Li et al. [20]. High-frequency bytes such as 0xE8/0xFF (common x86 opcodes) dominate their corresponding rows and columns, alongside the null byte (0x00) and the printable-ASCII range (0x20–0x7E) originating from strings; Figure 1(c) illustrates how these bytes lead to a common structure. Obfuscation flattens the distribution and introduces noise, and code-stripping packers such as ZIP and zlib reduce the image to near-pure noise; any bytes the model might attend to within this structure would be too fine for HiResCAM to resolve. Band ablation confirms the breadth of attention, degrading accuracy by at most 0.02 across all packers. Markov discards layout, so it cannot rely on headers or padding; equally, we cannot say what it does rely on.

The heatmaps from **Grayscale** show broad focus on the global image, except under the executable-focused packers UPX and MPRESS, where attention concentrates on the unencrypted code stubs appended to the bottom of the binary by both packers (Figure 4); [7] attests that the CNN's attention shifts to the stub under UPX packing. Ablation, however, attributes only minor importance to this region: occluding the bottom 32 rows costs MPRESS 0.15, while UPX shows negligible degradation (≤ 0.01). Combined with this broadly insignificant degradation (MPRESS aside), we conclude that *Grayscale-based classification does not rely on header or padding information*, but on the global image, which captures the overall binary layout and general visual patterns, in line with [1]. This contradicts [7], whose heatmaps map onto specific code regions, especially within particular families; this discrepancy could point to near-duplicate memorization in our dataset, but the leakage-free MPRESS case (0.965 accuracy) dismisses that explanation entirely. This is an advantage our study carries over prior explainability work, which does not experiment with a wide variety of packers. The Grayscale picture nonetheless remains inconclusive.

The heatmaps from **Entropy** exhibit the most intriguing behavior. The high deviation between family heatmaps is the strongest evidence in this study of learned family-discriminative structure, and correlates closely with the encoding's near-perfect accuracies. Band ablation does however raise a concern: under zlib and ZIP, which all other encodings fall short on, top *and* bottom occlusion each collapse Entropy to random guessing probabilities. If interpreted as causality, this result is paradoxical since the model cannot depend fully on the header and padding simultaneously. ZIP and zlib headers are at most 30 and 4 bytes [21; 22], which influence at most two and one pixels of the encoded image respectively, being followed by high-entropy compressed data immediately afterwards. This is far too little data to classify the families. The likeliest explanation is therefore not header reliance but the generation of out-of-distribution inputs under occlusion. As Figure 5 shows, zlib and ZIP are the only packers whose entropy output is an entirely high byte-stream, so during training the model never sees dark pixels except from the padding, which is minimal in height. This also explains the more gradual collapse under bottom-row occlusion, as the masked rows become increasingly bigger than the usual padding.

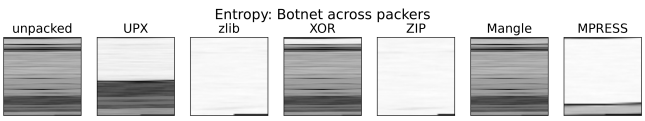


Figure 5: Entropy-encoded Botnet samples across different packers

To further investigate the out-of-distribution hypothesis directly, we removed the padding outright rather than masking it: the bottom 10 pixels were cropped and the image resized back to 224×224. The model was then *re-trained from scratch*. Performance was retained across every packer (Table 6), including the zlib (0.996) and ZIP (0.998) cases that collapse to random under band occlusion. This confirms the collapse is caused by out-of-distribution samples, and rules out the padding as a discriminative feature for Entropy.

Band occlusion is therefore an invalid explanation of the CNN's decisions. Combined with the header argument above, the trim experiment narrows **RQ3** for Entropy: its decision making lies neither in the headers nor the padding, but in the actual high-entropy body of the compressed content. Precise locations of interest within that body however remain unre-

Table 6: Entropy accuracy and macro-F1 with the bottom 10 pixels (padding) cropped and the image resized back to 224×224.

	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
Acc.	1.000	0.998	0.996	1.000	0.998	1.000	0.990
F1	1.000	0.998	0.996	1.000	0.998	1.000	0.991

solved.

5.3 Limitations

Several limitations qualify these findings.

Dataset Limitations

The synthetically-generated dataset was provided under confidentiality by Akash Amalan; as such, *the malware binaries cannot be published*. The origin is undisclosed and we cannot characterize how the packed variants were generated; we must reason only from properties that can be measured directly. Synthetic generation of variants may also misrepresent similarity between same-family variants in the real world and shortcuts we identify could be over-pronounced compared to real malware.

Furthermore, the 54.5% TLSH leakage on XOR evidences erroneous packing. Comparison of the byteplots with the unpacked versions shows that the majority of structural information is preserved, unlike the expected high entropy output. **No conclusions can be drawn from models trained on XOR.**

Explainability Limitations

Band ablation gives limited structural interpretation. A more thorough explainability analysis would consider occlusion of specific sections and elements of the images rather than just general headers and padding. Furthermore, out-of-distribution samples lead to problematic readings; ablation before training should be considered in more comprehensive studies. Our ablation baseline is computed from 40 samples per family, some of which the model may have seen in training; this inflates the absolute baseline but not our analysis, which focuses only on relative degradation.

HiResCAM is insufficient. All heatmaps are upsampled from a base resolution of 7×7, a limitation imposed by our CNN architecture. This results in visualizations that are too coarse to interpret in depth.

Resizing

Information loss during resizing is disproportionate across the three encodings. Markov suffers constant, minimal loss. Entropy loses a considerable amount depending on the binary size, and Grayscale in the order of $16\times$ that of Entropy, given our stride length. This limitation cannot be mitigated due to a controlled classifier architecture.

6 Conclusions and Future Work

Our study presents the first controlled, head-to-head comparison of four binary-to-image encodings - grayscale and RGB byteplots, Markov bigram plots, and sliding-window Shannon entropy - under a fixed ResNet-18 classifier and a balanced dataset spanning seven packer conditions. The central finding is that these encodings cannot be ranked by raw accuracy alone: doing so conflates models that learn genuine malware structure with those that exploit shortcuts.

We answer our three research questions directly. **RQ1**: unpacked family classification is trivial across all encodings which achieve perfect accuracy, but the encodings diverge sharply under obfuscation. Entropy dominates the other encodings, while the byteplots perform equivalently and moderately outperform Markov. Entropy is uniquely resilient, retaining near-perfect accuracy even under compression, where the others collapse. **RQ2**: file-size is an easily exploitable shortcut that alone yields near-perfect accuracy on most packers, yet the encodings do not rely on it totally. Markov is categorically file-size invariant; under MPRESS the size shortcut is largely neutralized while accuracy is retained across all non-Markov encoders. All encodings also outperform the entropy baseline on at least three packers, and TLSH leakage cannot explain high accuracies on MPRESS. Shortcuts therefore complement, rather than replace, genuinely learned family features. **RQ3**: although our attention tools were unable to pinpoint the decisive regions of the models, they rule out reliance on headers and padding for all encodings, and attribute Entropy's decision making to its high-entropy content body. Furthermore, Entropy's family-distinct heatmaps provide evidence of discriminative family-structure learning, even if exact attention attribution remains unknown.

Beyond the comparison itself, our methodological contribution is the demonstration that no single diagnostic tool suffices: the random-forest baselines, TLSH analysis, and ablation each miss what the others catch. Only by reasoning about them as a whole could we separate shortcut-driven accuracies from genuine learning. Two results stand out as warranting further study: Entropy's resilience under obfuscation, and the MPRESS packer, in which our framework finds no shortcut capable of explaining the high accuracy achieved.

6.1 Future Work

Real-world data. Our dataset's strength is its breadth of packers. Its weakness is that synthetically generated same-family variants may exaggerate the shortcuts we measure, especially regarding file-size. Repeating the study on non-synthetic malware, ideally re-packed with the same methods covered in this study, would test whether the encodings still recover family structure when those shortcuts are weaker.

A dedicated explainability study of Entropy. Given its standout performance, Entropy merits deeper analysis than a single CAM method allows. Section-level occlusion applied at training time (e.g. removing headers or the .mpress1 section rather than fixed image bands) would avoid the out-of-distribution pitfalls that invalidated our inference-time ablation, and provide more focused insights into attention attribution. Complementing HiResCAM with other attribution methods such as SHAP and LIME [6; 7] would strengthen the attention findings and may help close the gap left by **RQ3**.

Higher-resolution attention. The 7×7 final convolutional layer of ResNet-18 results in coarse heatmaps; a larger input resolution would yield finer maps able to resolve smaller structural features that our current maps cannot.

7 Responsible Research

Our research aims to reinforce and enrich scientific knowledge in a responsible manner. For full transparency and confirmation of findings, reproducibility is ensured to the maximum possible extent (Section 7.1). Given the nature of our work in the cybersecurity field, ethical concerns are raised and discussed (Section 7.2). Lastly, the use and applications of generative AI throughout the course of this study are detailed (Section 7.3).

7.1 Reproducibility

For reasons discussed in Section 5.3, the original dataset consisting of malware binaries cannot be published. As remediation, all encoded malware images that were used in training and evaluation are released in the linked GitHub repository[3], alongside all the results, code, saved models, and Jupyter notebooks used. The practical implications on reproducibility are that the full methodology in Section 3 can be reproduced, with the exception of the actual image encoding step.

7.2 Ethics

Two main ethical issues arise from the nature of the research conducted in this paper. Both issues are evaluated and mitigating actions are described if applicable.

Distribution of Malware Binaries

The research was conducted on a dataset containing unpacked, as well as obfuscated malware binaries. The concern of malicious software execution was mitigated: the malware binaries were neutered, meaning they are unable to be executed, as disclosed by Akash Amalan. Furthermore, the dataset was distributed amongst internal group members solely through SURFdrive, which restricts access to those with institutional access. The binaries are not publicly published in this report and only the encoded, already resized images are made available; information loss during resizing makes reconstruction of the binaries infeasible. Therefore *contagion and execution of harmful malware is not possible as a result of this study*.

Adversarial Considerations

As a malware classification study, this paper carries intrinsic value to malicious parties who may seek to evade image-based detection systems. The shortcut analysis performed in the study highlighted file-size and entropy distribution statistics as potentially useful signals in CNN-based malware detection. The concern of adversaries exploiting this knowledge is weighed by two mitigating factors. File-size and global entropy are features that are already in use by signature-based detectors; furthermore, file-size inflation is a well-established obfuscation method [23], and global entropy changes are a direct result of many existing packers. As such, no valuable information is gained by the adversaries outside of what is already available in the public domain.

Lastly, we discuss certain (encoding × packer) pairs that perform poorly, revealing offensive knowledge that can be used to inform packer selection when targeting specific encodings. This concern is weighed by two mitigating factors. Our findings are limited to our synthetic dataset, so the adversary cannot guarantee that the same weaknesses extrapolate to real-world malware. Secondly, the revealed weaknesses present just as much an offensive opportunity to adversaries as they offer defensive knowledge required to build detectors that do not suffer from the same weaknesses. It is therefore argued that the defensive value of knowing where these encodings fail outweighs the limited and largely pre-existing offensive value of the findings.

[3] https://github.com/MartimCardeira/MalwareImagesRP

7.3 Use of Generative AI

Throughout the course of this study, generative AI tools were used **solely for the following purposes**:

- **Code Assistance**: Claude was used mostly for the creation of boilerplate code. Method signatures and docstrings were provided as context in prompts for the generation of trivial experiment loops. Most code, both hand-written and AI generated was also passed to Claude to check for code correctness, complementing manual inspection. This was done to ensure the code functioned as intended before committing large Slurm jobs with long wait times to the Delft AI Cluster (DAIC).
- **Data Visualization**: Creation of tables in Overleaf, as well as plots made in Matplotlib were largely handled by Claude for efficiency. All outputs were scrutinized to make sure they were congruent with the numerical results. In the case of Matplotlib, the respective code was also examined for correctness.
- **Language support**: Drafts for content heavy sections such as the Discussion section were first written by hand, and then given as context to Claude to reformulate them in a more concise manner. This was done for the purposes of fulfilling the page limit and improving the reading experience.

Note on data privacy. At no point were any malware binaries nor encoded images of the confidential dataset shared with Claude or any other LLMs. Context of the dataset was merely limited to numerical results for visualization, which do not reveal the actual underlying data.

References

[1] 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*, (Pittsburgh Pennsylvania USA), pp. 1–7, ACM, July 2011.

[2] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, p. 107138, Apr. 2020.

[3] W. Nie and C. Zhu, "γ-M2I: Image-based malware classification via feature spatial transformation," *Journal of Information Security and Applications*, vol. 97, p. 104355, Mar. 2026.

[4] H. J. Darton, "Malware Detection with CNNs on Entropy and Greyscale Images," *Latin-American Journal of Computing*, vol. 13, pp. 45–53, Jan. 2026.

[5] 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, June 2025.

[6] S. Nazim, M. M. Alam, S. S. Rizvi, J. C. Mustapha, S. S. Hussain, and M. 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, p. e0318542, May 2025.

[7] M. Brosolo, V. Puthuvath, and M. Conti, "The Road Less Traveled: Investigating Robustness and Explainability in CNN Malware Detection," Mar. 2025. arXiv:2503.01391 [cs.CR].

[8] D. Gibert, N. Totosis, C. Patsakis, Q. Le, and G. Zizzo, "Assessing the impact of packing on static machine learning-based malware detection and classification systems," *Computers & Security*, vol. 156, p. 104495, Sept. 2025.

[9] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3971–3988, 2022.

[10] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, pp. 665–673, Nov. 2020.

[11] M. Brosolo, A. K. A, M. Conti, R. R. K. A, M. S. K. P, S. Nicolazzo, A. Nocera, and V. P, "Security through the eyes of AI: How visualization is shaping malware detection," *Computer Science Review*, vol. 61, p. 100914, 2026.

[12] M. Brosolo, V. Puthuvath, A. KA, R. Rehiman, and M. Conti, "SoK: Visualization-based Malware Detection Techniques," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ARES '24, (New York, NY, USA), pp. 1–13, Association for Computing Machinery, July 2024.

[13] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on markov images and deep learning," *Computers & Security*, vol. 92, p. 101740, May 2020.

[14] R. Lyda and J. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *IEEE Security & Privacy*, vol. 5, pp. 40–45, Mar. 2007.

[15] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security*, vol. 14, pp. 1–14, Feb. 2015.

[16] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Classification of Malware by Using Structural Entropy on Convolutional Neural Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, Apr. 2018.

[17] S. Dambra, Y. Han, S. Aonzo, P. Kotzias, A. Vitale, J. Caballero, D. Balzarotti, and L. Bilge, "Decoding the Secrets of Machine Learning in Malware Classification: A Deep Dive into Datasets, Feature Extraction, and Model Performance," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, (Copenhagen Denmark), pp. 60–74, ACM, Nov. 2023.

[18] J. Oliver, C. Cheng, and Y. Chen, "TLSH – A Locality Sensitive Hash," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 7–13, Nov. 2013.

[19] R. L. Draelos and L. Carin, "Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks," Nov. 2021. arXiv:2011.08891 [eess.IV].

[20] W.-J. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints: identifying file types by n-gram analysis," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pp. 64–71, June 2005.

[21] PKWARE, Inc., ".ZIP File Format Specification, Version 6.3.9," July 2020. https://pkwaredownloads.blob.core.windows.net/pkware-general/Documentation/APPNOTE-6.3.9.TXT.

[22] L. P. Deutsch and J.-l. Gailly, "ZLIB Compressed Data Format Specification version 3.3," Request for Comments RFC 1950, Internet Engineering Task Force, May 1996.

[23] MITRE, "Obfuscated Files or Information: Binary Padding, Sub-technique T1027.001," May 2026. https://attack.mitre.org/techniques/T1027/001/.

A Hyperparameter Tuning

Markov

Nie and Zhu [3] state that the choice of hyperparameter γ is dataset dependent; they pick $\gamma = 0.5$ as the overall best value based on their testing [3]. As we work with a different dataset, we must determine the best γ empirically through experimentation. A set of γ values to try was determined trivially: (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0). The value $\gamma = 1.0$ is equivalent to naive Markov without the gamma mapping [13], hence it serves as both the upper bound and baseline. $\gamma = 0.0$ maps all pixels to maximum intensity and is a degenerate case, hence $\gamma = 0.1$ is the lower bound.

The Markov method was used to encode the entire dataset ten different times with the aforementioned γ values. Figure 6 illustrates the differences between some of the tested γ values on the same exact unpacked *Adware* binary.

(a) $\gamma = 0.1$ (b) $\gamma = 0.4$ (c) $\gamma = 0.7$ (d) $\gamma = 1.0$

Figure 6: Markov bigram plots of the same *Adware* sample at gamma values 0.1, 0.4, 0.7, and 1.0.

The result is a clear trade-off between visibility and accurate transition representations. At the extreme end of $\gamma = 0.1$, very rare transitions have their pixel intensities inflated, making them visible in the image and preventing truncation. At the other end, $\gamma = 1.0$ (naive Markov) exhibits the exact truncation problem that Nie and Zhu [3] aimed to solve with gamma mapping. With all image encodings processed, we ran the full training/evaluation pipeline from Section 3.4, restricted to just these 10 encodings. The trend-line for accuracy vs γ is shown in Figure 7, while a more complete bar graph depicting accuracy and macro-F1 scores is shown in Figure 8. Tables 7 and 8 display the respective values.

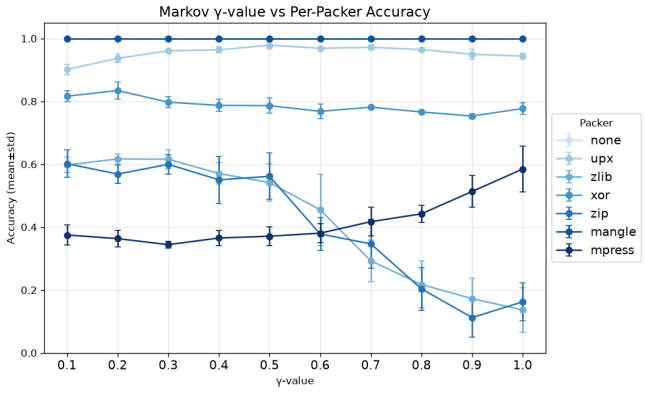


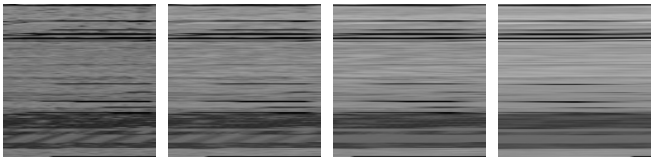
Figure 7: Markov encoding accuracy across gamma values, broken down by packer (5-fold CV, error bars show SD).

The results show varying trends between packers and γ values. ZIP and zlib (high entropy compression methods) demonstrate extremely high performance degradation past $\gamma = 0.5$. MPRESS is the only packer to experience a strong positive accuracy correlation with γ. The remaining packers exhibit some slight fluctuations, although their highest mean accuracy appears to be around $\gamma = 0.5$. Based on this evidence, we adopt the $\gamma = 0.5$ version of the Markov encoding for our study as it exhibits the best overall performance.

Sliding Window Shannon Entropy

The Sliding Window Shannon entropy method has two parameters: Window size and stride. An initial baseline window size of 32 bytes was chosen as used by [4]. As file-sizes in the dataset are $\geq$ 25KB, and sections in binaries are much greater than 32 bytes, this was reasoned as the lower bound of window sizes to test. Stride length was not detailed, although a stride of 1 byte could be implied. In any case, initial experimentation showed that a stride of 1 would be computationally intractable in the given time frame. A stride of 16 bytes was found to be appropriate, ensuring speed during encoding while having windows overlap by at least 50%, even on the smallest tested window size. No further stride lengths were tested; small sample-size tests revealed negligible differences with higher stride lengths, especially on higher window sizes.

The Shannon entropy method was used to encode the entire dataset four different times, with window sizes of (32, 64, 128, 256) bytes, and the fixed stride value of 16 bytes. Figure 9 illustrates the differences between the tested window sizes on the same exact unpacked *Adware* binary.



(a) 32 bytes (b) 64 bytes (c) 128 bytes (d) 256 bytes

Figure 9: Sliding-window Shannon entropy encodings of the same *Adware* sample at window sizes 32, 64, 128, and 256 bytes (stride of 16 bytes).

The result is increasing image smoothness and delineation of low/high entropy sections as window size increases, with decreasing granularity. With all image encodings processed, we ran the full training/evaluation pipeline from Section 3.4, restricted to just the 4 encodings. The results for accuracy and macro-F1 are shown in Figure 10. Tables 9 and 10 display the respective values for each.

The results show very high accuracies ($\geq$ 0.975) across all packers and window sizes. A general trend of slight performance degradation and greater spread can be observed as window size is decreased. As the window size of 256 bytes offers the best overall performance and near-perfect accuracies across all packers, we adopt the 256-byte version of Entropy for our study.

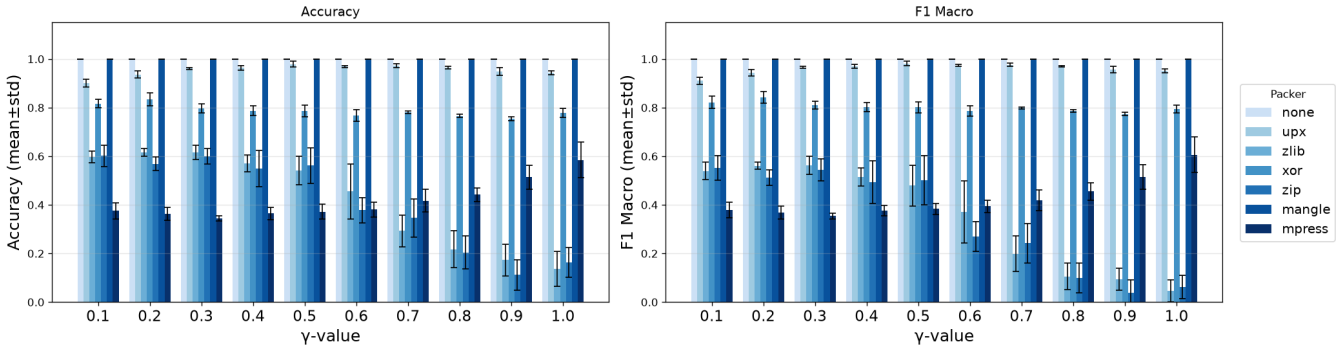


Figure 8: Accuracy and macro-F1 of the Markov encoding across gamma values, per packer (5-fold CV, error bars show SD).

Table 7: Markov encoding accuracy (mean±std, 5-fold CV) across gamma values and packers.

γ	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
0.1	1.000±.000	0.902±.016	0.598±.024	0.816±.018	0.602±.043	1.000±.000	0.375±.033
0.2	1.000±.000	0.937±.014	0.617±.016	0.834±.027	0.569±.029	1.000±.000	0.363±.027
0.3	1.000±.000	0.961±.004	0.616±.030	0.797±.018	0.600±.031	1.000±.000	0.344±.011
0.4	1.000±.000	0.964±.010	0.571±.034	0.787±.020	0.551±.075	1.000±.000	0.365±.025
0.5	1.000±.000	0.979±.012	0.542±.059	0.786±.024	0.562±.074	1.000±.000	0.371±.031
0.6	1.000±.000	0.969±.004	0.455±.113	0.768±.024	0.378±.052	1.000±.000	0.381±.030
0.7	1.000±.000	0.972±.008	0.293±.066	0.781±.004	0.347±.079	1.000±.000	0.417±.046
0.8	1.000±.000	0.965±.005	0.218±.075	0.766±.006	0.204±.068	1.000±.000	0.442±.028
0.9	1.000±.000	0.950±.016	0.173±.066	0.753±.008	0.112±.062	1.000±.000	0.513±.050
1.0	1.000±.000	0.944±.009	0.137±.071	0.777±.019	0.162±.061	1.000±.000	0.585±.074

Table 8: Markov encoding macro-F1 (mean±std, 5-fold CV) across gamma values and packers.

γ	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
0.1	1.000±.000	0.910±.015	0.539±.037	0.822±.026	0.552±.050	1.000±.000	0.379±.033
0.2	1.000±.000	0.944±.013	0.561±.015	0.842±.024	0.512±.031	1.000±.000	0.369±.027
0.3	1.000±.000	0.966±.004	0.563±.037	0.810±.015	0.545±.046	1.000±.000	0.354±.012
0.4	1.000±.000	0.969±.008	0.514±.037	0.803±.018	0.495±.088	1.000±.000	0.376±.021
0.5	1.000±.000	0.982±.010	0.479±.085	0.801±.022	0.502±.100	1.000±.000	0.384±.023
0.6	1.000±.000	0.973±.003	0.371±.128	0.786±.021	0.269±.061	1.000±.000	0.394±.026
0.7	1.000±.000	0.976±.007	0.199±.074	0.799±.004	0.242±.081	1.000±.000	0.419±.042
0.8	1.000±.000	0.970±.004	0.106±.055	0.787±.005	0.099±.061	1.000±.000	0.455±.036
0.9	1.000±.000	0.957±.013	0.094±.046	0.773±.007	0.037±.053	1.000±.000	0.514±.051
1.0	1.000±.000	0.952±.007	0.047±.045	0.795±.015	0.063±.048	1.000±.000	0.606±.073

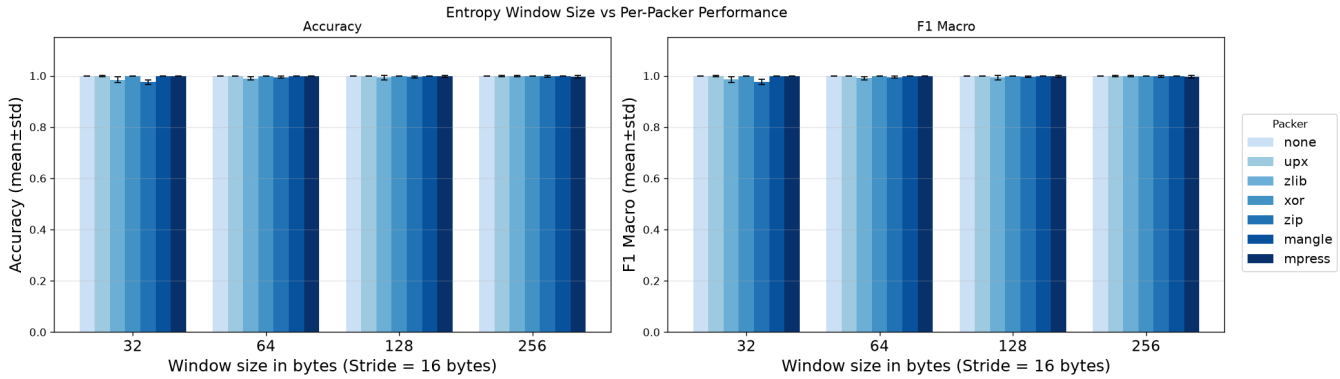


Figure 10: Accuracy and macro-F1 of Entropy across the four tested window sizes (32, 64, 128, 256 bytes; stride 16 bytes), 5-fold CV, error bars show SD.

Table 9: Entropy accuracy (mean±std, 5-fold CV) across window sizes and packers. Stride fixed at 16 bytes; window size in bytes.

Window	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
32	1.000±0.000	0.999±0.002	0.985±0.011	1.000±0.000	0.975±0.009	1.000±0.000	1.000±0.000
64	1.000±0.000	1.000±0.000	0.990±0.006	1.000±0.000	0.994±0.004	1.000±0.000	1.000±0.000
128	1.000±0.000	1.000±0.000	0.993±0.009	1.000±0.000	0.996±0.004	1.000±0.000	0.998±0.004
256	1.000±0.000	0.999±0.002	0.999±0.002	1.000±0.000	0.998±0.004	1.000±0.000	0.996±0.005

Table 10: Entropy macro-F1 (mean±std, 5-fold CV) across window sizes and packers. Stride fixed at 16 bytes; window size in bytes.

Window	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
32	1.000±0.000	0.999±0.002	0.985±0.011	1.000±0.000	0.976±0.010	1.000±0.000	1.000±0.000
64	1.000±0.000	1.000±0.000	0.991±0.006	1.000±0.000	0.995±0.003	1.000±0.000	1.000±0.000
128	1.000±0.000	1.000±0.000	0.993±0.008	1.000±0.000	0.996±0.003	1.000±0.000	0.998±0.003
256	1.000±0.000	0.999±0.002	0.999±0.002	1.000±0.000	0.998±0.004	1.000±0.000	0.996±0.005

B Additional Tables and Figures

Table 11: Image width for various file-sizes, following [1]

File-size Range	Image Width
$<$10 kB	32
10 kB – 30 kB	64
30 kB – 60 kB	128
60 kB – 100 kB	256
100 kB – 200 kB	384
200 kB – 500 kB	512
500 kB – 1000 kB	768
$>$1000 kB	1024

Table 12: (Encoding × packer) test accuracy (mean ± std, 5-fold CV).

Enc.	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
Grayscale	$1.000_{\pm.000}$	$0.872_{\pm.010}$	$0.496_{\pm.026}$	$0.993_{\pm.002}$	$0.543_{\pm.029}$	$1.000_{\pm.000}$	$0.965_{\pm.019}$
RGB	$1.000_{\pm.000}$	$0.855_{\pm.022}$	$0.551_{\pm.021}$	$0.990_{\pm.009}$	$0.545_{\pm.026}$	$1.000_{\pm.000}$	$0.933_{\pm.030}$
Markov	$1.000_{\pm.000}$	$0.979_{\pm.012}$	$0.542_{\pm.059}$	$0.786_{\pm.024}$	$0.562_{\pm.074}$	$1.000_{\pm.000}$	$0.371_{\pm.031}$
Entropy	$1.000_{\pm.000}$	$0.999_{\pm.002}$	$0.999_{\pm.002}$	$1.000_{\pm.000}$	$0.998_{\pm.004}$	$1.000_{\pm.000}$	$0.996_{\pm.005}$

Table 13: (Encoding × packer) test macro-F1 (mean ± std, 5-fold CV).

Enc.	none	UPX	zlib	XOR	ZIP	Mangle	MPRESS
Grayscale	$1.000_{\pm.000}$	$0.886_{\pm.009}$	$0.466_{\pm.022}$	$0.994_{\pm.002}$	$0.523_{\pm.029}$	$1.000_{\pm.000}$	$0.967_{\pm.018}$
RGB	$1.000_{\pm.000}$	$0.872_{\pm.020}$	$0.522_{\pm.017}$	$0.991_{\pm.008}$	$0.525_{\pm.030}$	$1.000_{\pm.000}$	$0.935_{\pm.028}$
Markov	$1.000_{\pm.000}$	$0.982_{\pm.010}$	$0.479_{\pm.085}$	$0.801_{\pm.022}$	$0.502_{\pm.100}$	$1.000_{\pm.000}$	$0.384_{\pm.023}$
Entropy	$1.000_{\pm.000}$	$0.999_{\pm.002}$	$0.999_{\pm.002}$	$1.000_{\pm.000}$	$0.998_{\pm.004}$	$1.000_{\pm.000}$	$0.996_{\pm.005}$

Table 14: Markov band-ablation accuracy. Base = unoccluded baseline; T/B = top/bottom by band width (px).

Packer		Base	1	2	4	8	16	32
none, Mangle: baseline 1.00*,* 1.00 *at all bands (omitted)*								
UPX	T	1.00	.99	.99	1.00	.99	.99	.99
	B		1.00	1.00	1.00	1.00	.99	.99
zlib	T	.88	.88	.88	.89	.88	.89	.89
	B		.88	.88	.88	.88	.88	.92
XOR	T	.95	.95	.95	.95	.96	.95	.95
	B		.95	.95	.96	.95	.95	.95
ZIP	T	.92	.92	.92	.92	.92	.91	.91
	B		.92	.92	.92	.92	.91	.91
MPRESS	T	.87	.86	.86	.85	.85	.85	.87
	B		.87	.87	.88	.87	.86	.87

C HiResCAM Heatmaps

C.1 Grayscale

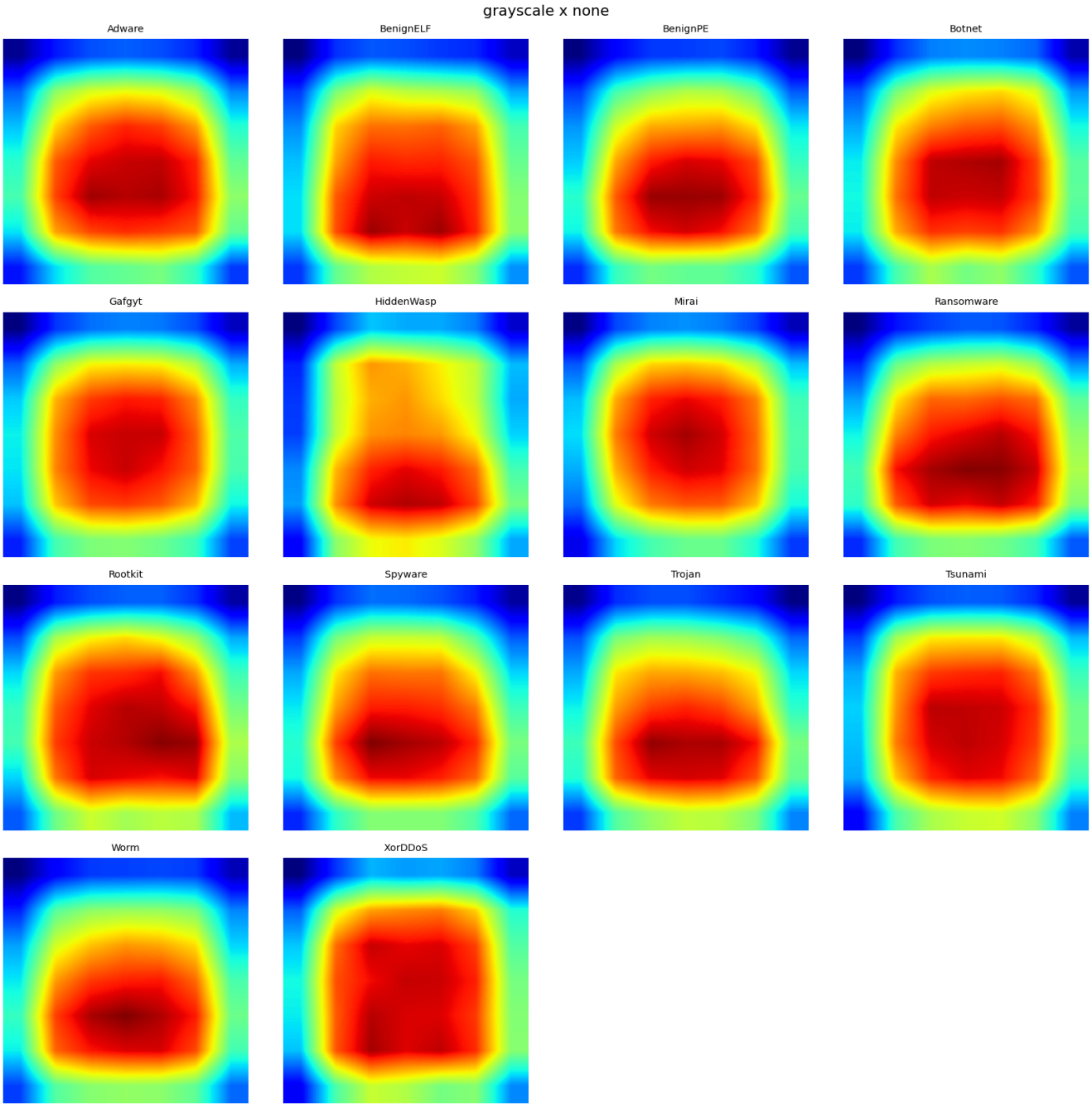


Figure 11: Cumulative class-level HiResCAM heatmaps for Grayscale under the none packer, shown per family.

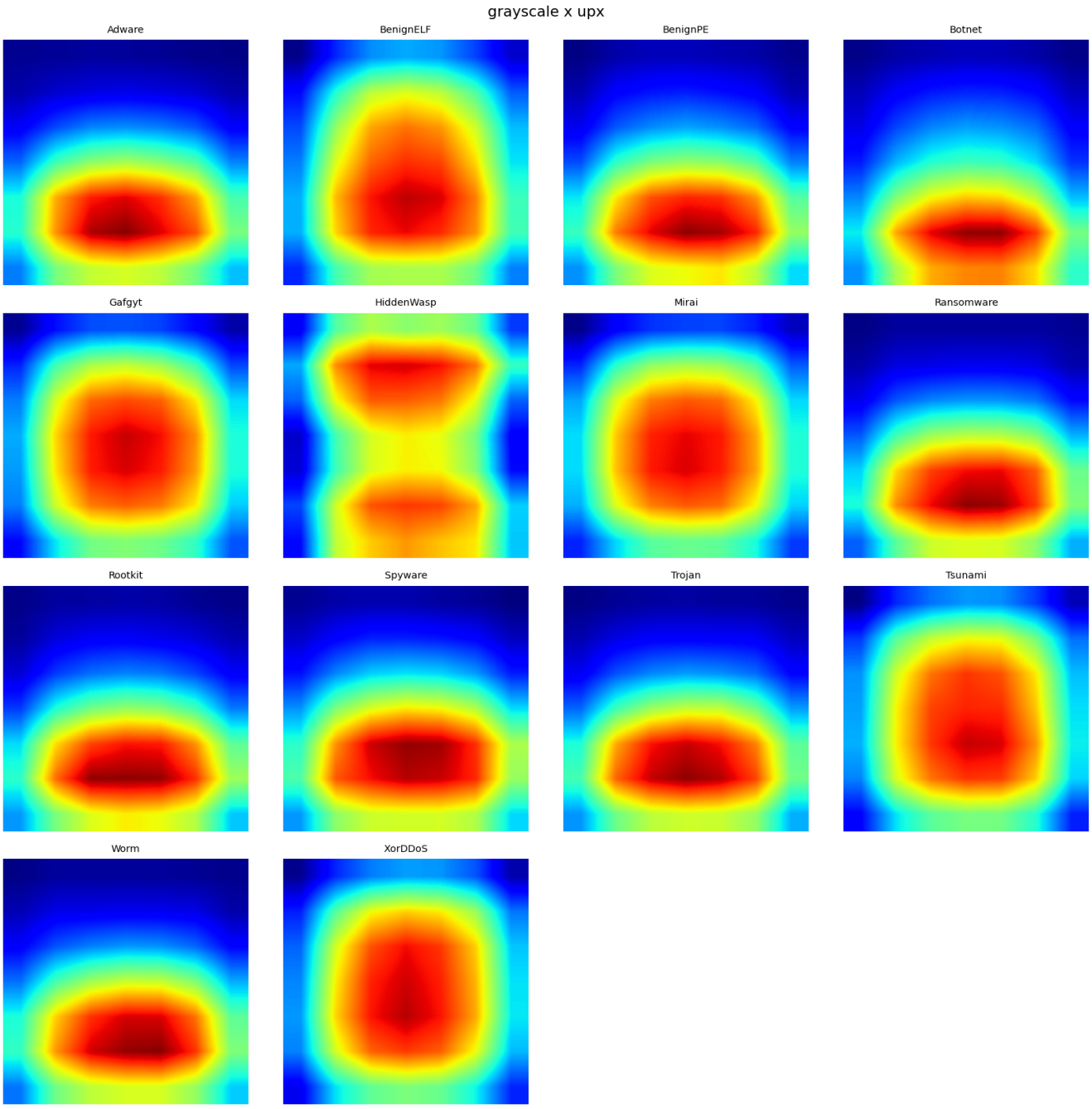


Figure 12: Cumulative class-level HiResCAM heatmaps for Grayscale under the UPX packer, shown per family.

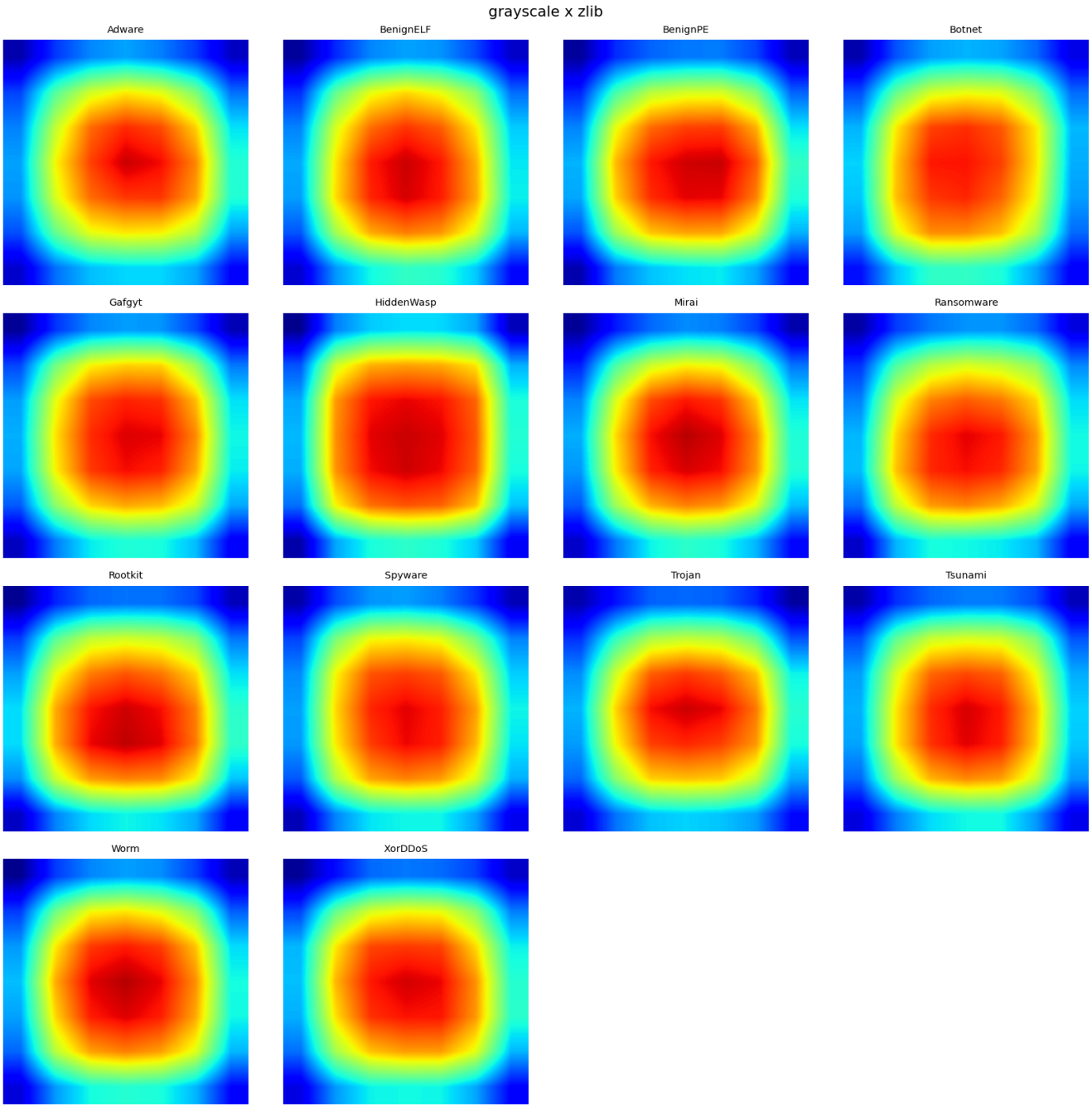


Figure 13: Cumulative class-level HiResCAM heatmaps for Grayscale under the zlib packer, shown per family.

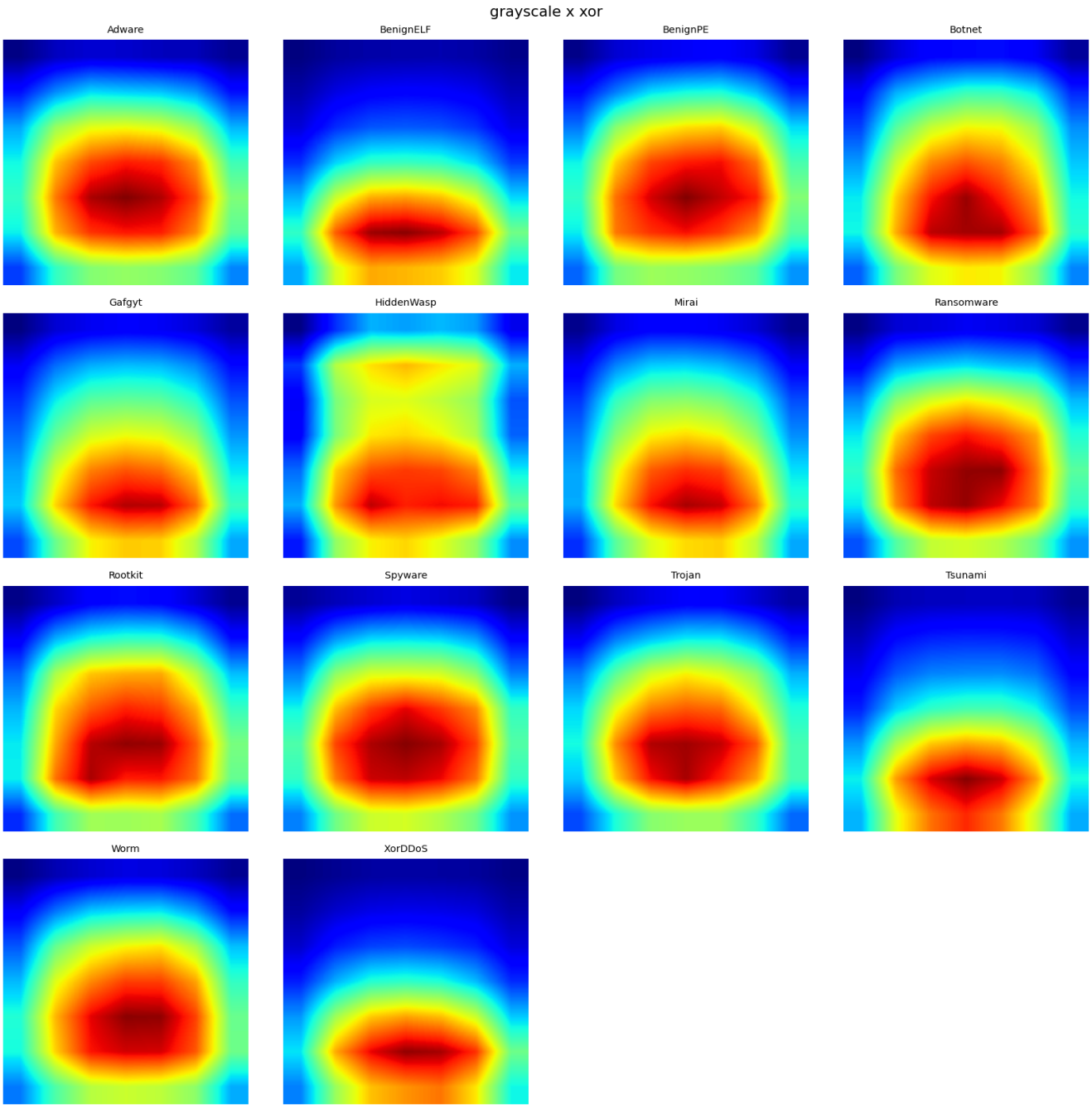


Figure 14: Cumulative class-level HiResCAM heatmaps for Grayscale under the XOR packer, shown per family.

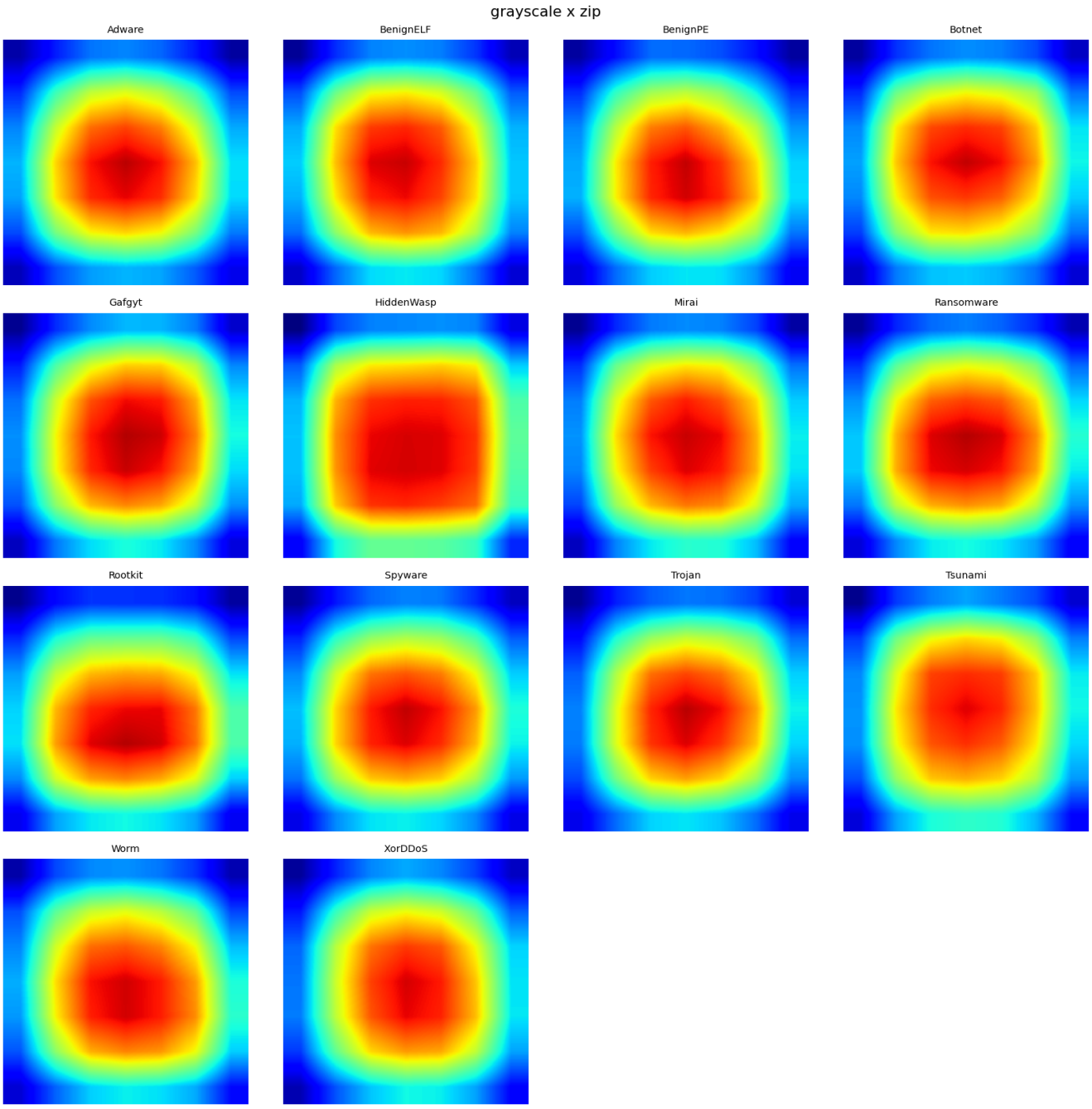


Figure 15: Cumulative class-level HiResCAM heatmaps for Grayscale under the ZIP packer, shown per family.

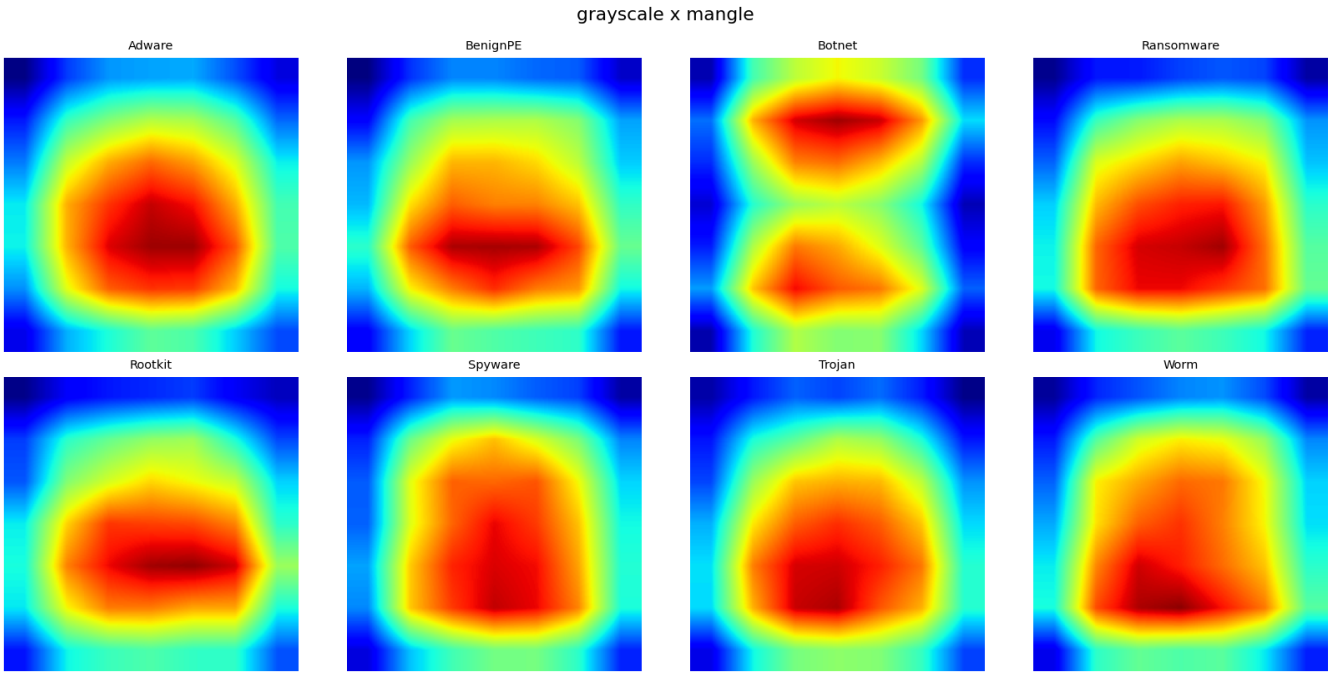


Figure 16: Cumulative class-level HiResCAM heatmaps for Grayscale under the Mangle packer, shown per family.

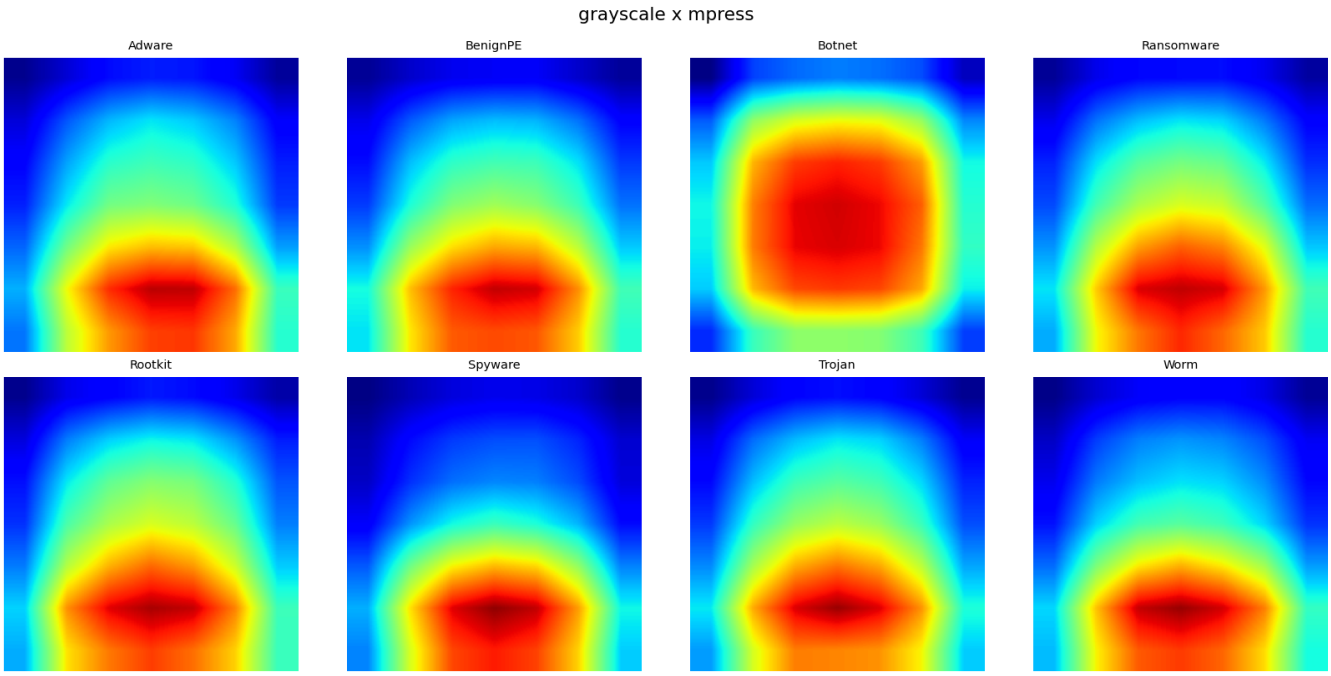


Figure 17: Cumulative class-level HiResCAM heatmaps for Grayscale under the MPRESS packer, shown per family.

C.2 Markov

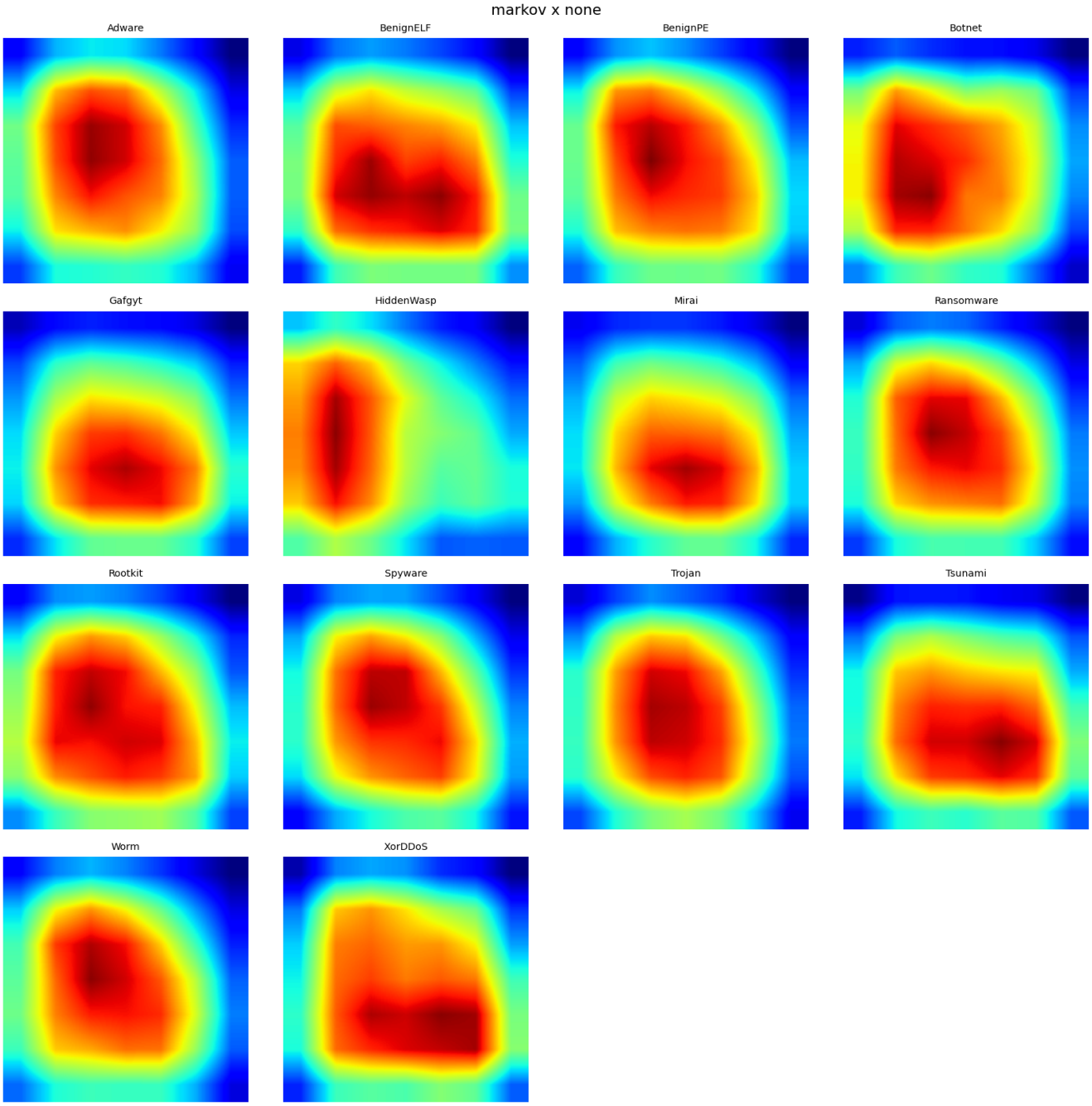


Figure 18: Cumulative class-level HiResCAM heatmaps for the markov encoding under the none packer, shown per family.

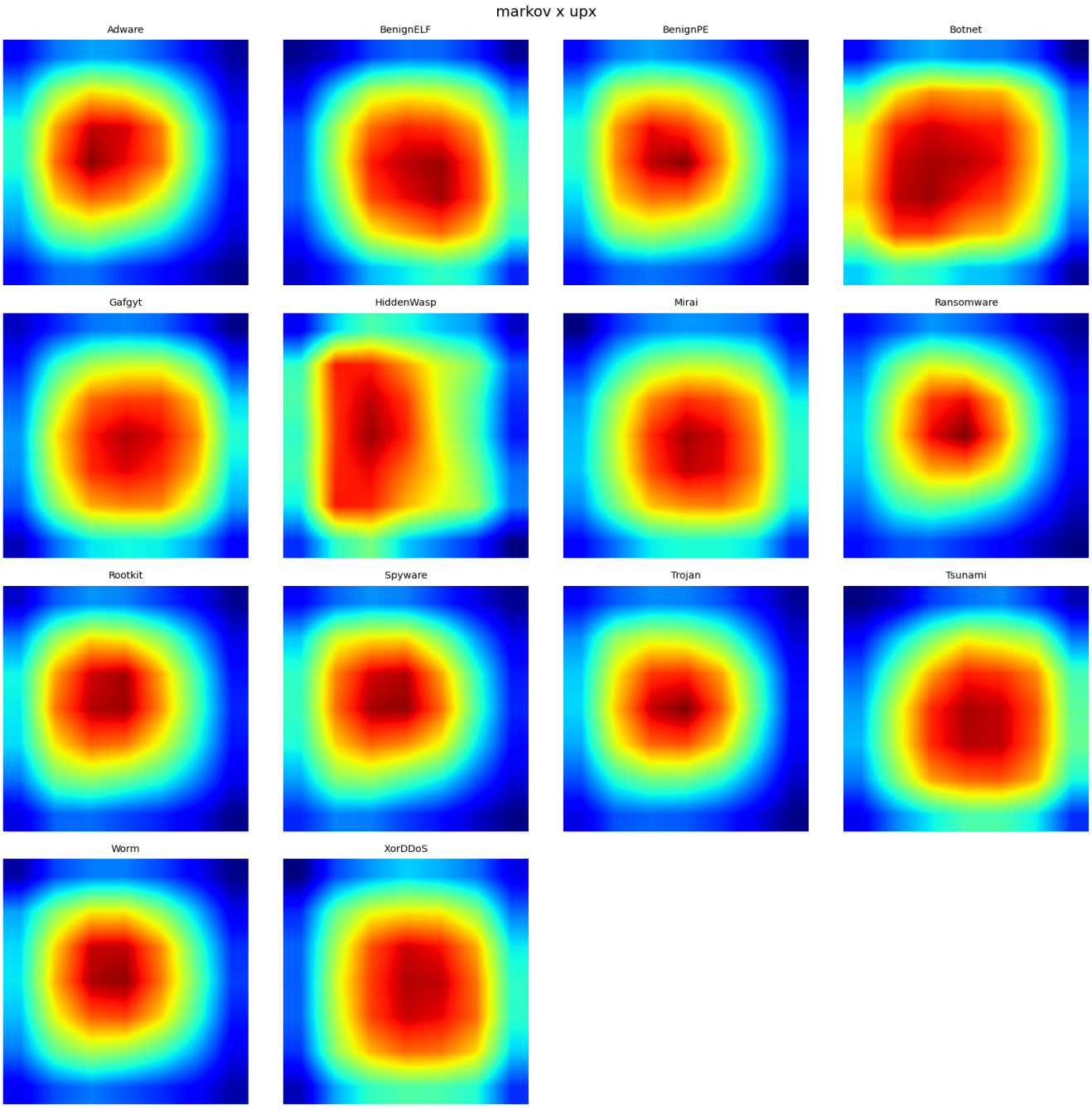


Figure 19: Cumulative class-level HiResCAM heatmaps for the markov encoding under the UPX packer, shown per family.

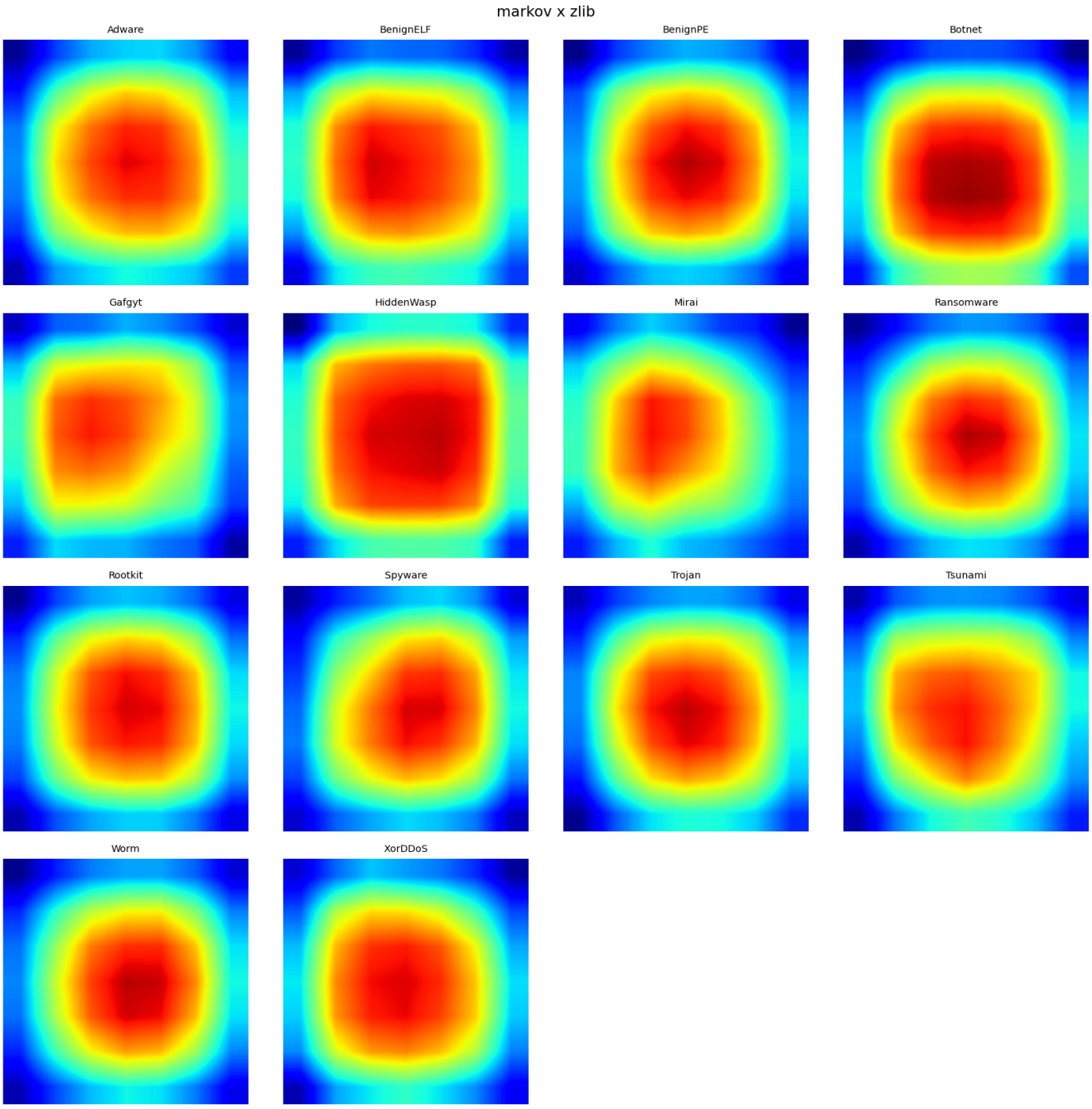


Figure 20: Cumulative class-level HiResCAM heatmaps for the markov encoding under the zlib packer, shown per family.

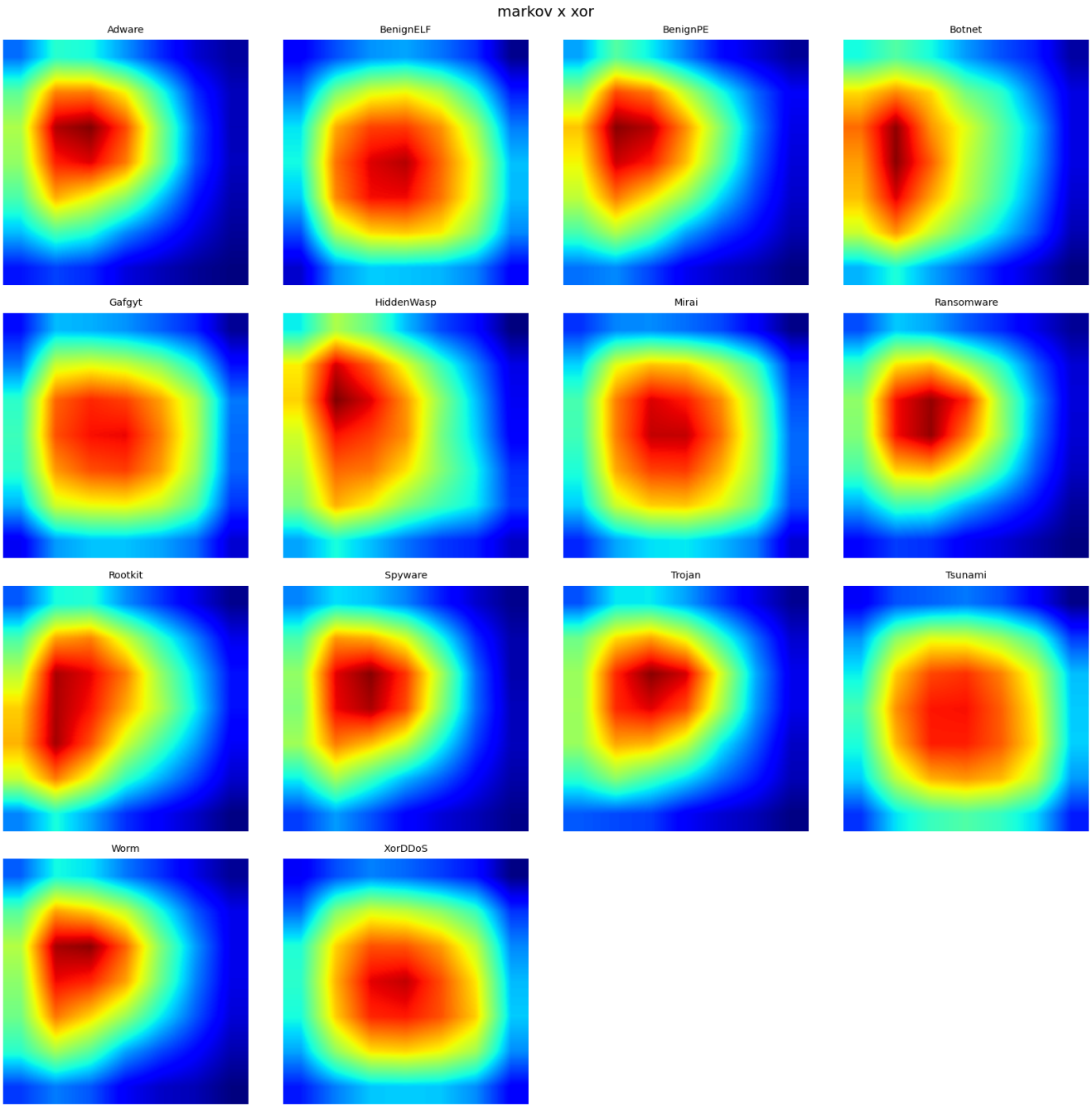


Figure 21: Cumulative class-level HiResCAM heatmaps for the markov encoding under the XOR packer, shown per family.

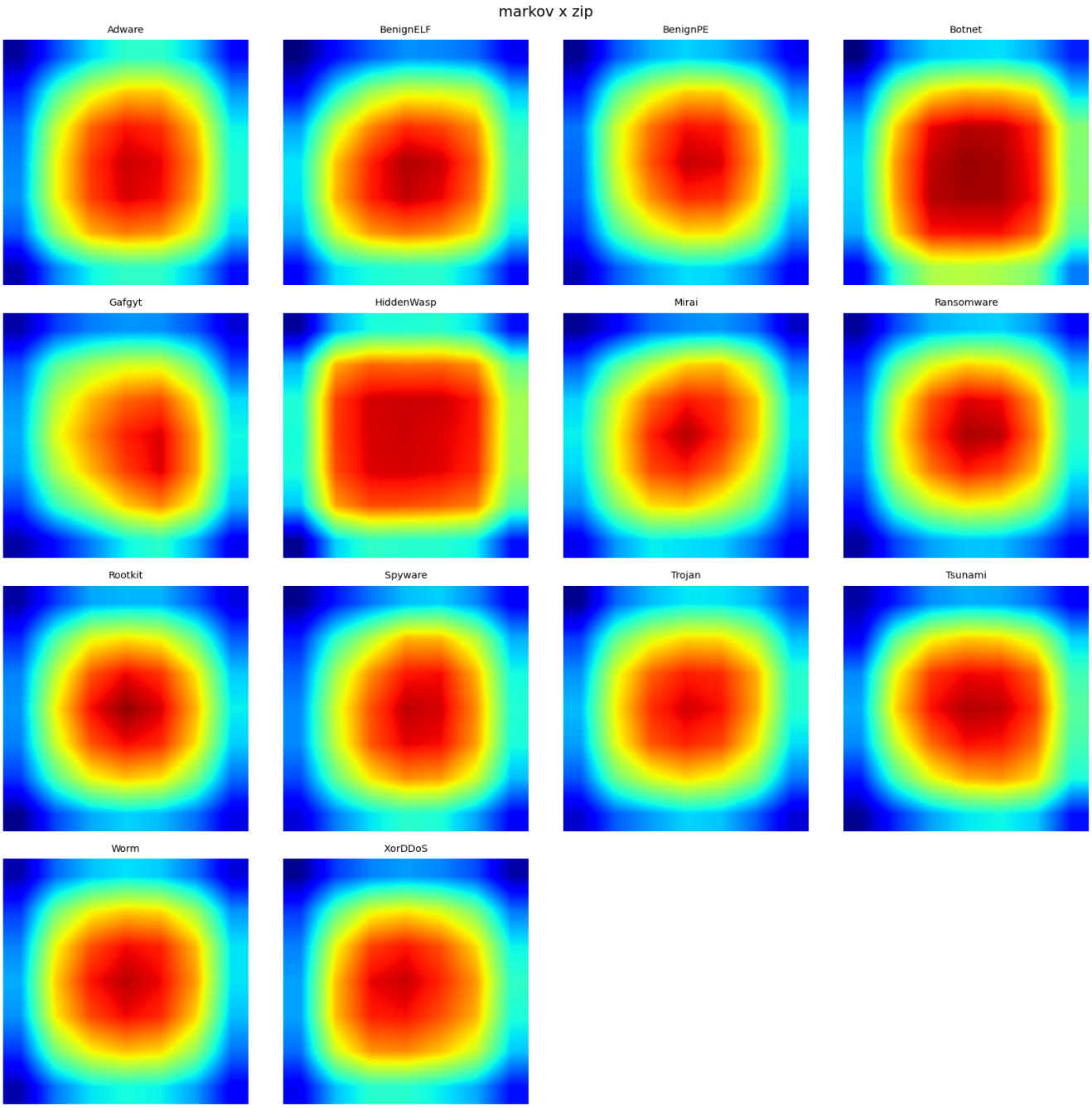


Figure 22: Cumulative class-level HiResCAM heatmaps for the markov encoding under the ZIP packer, shown per family.

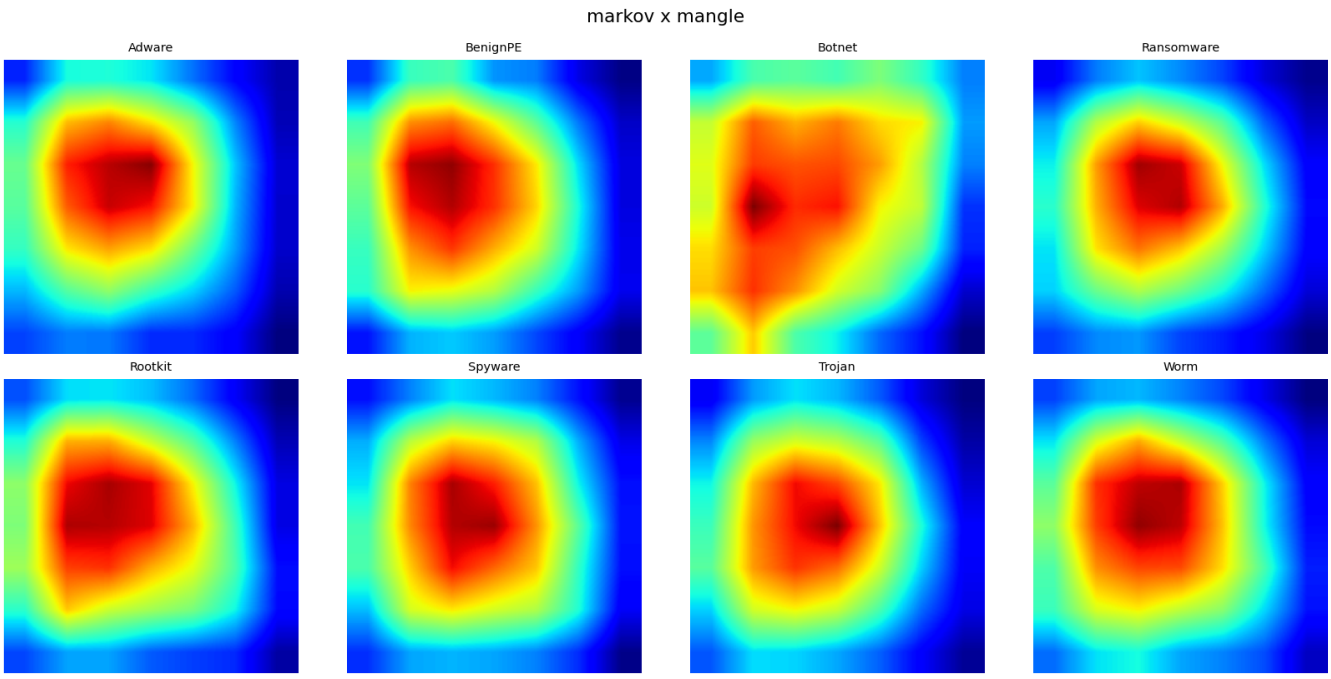


Figure 23: Cumulative class-level HiResCAM heatmaps for the markov encoding under the Mangle packer, shown per family.

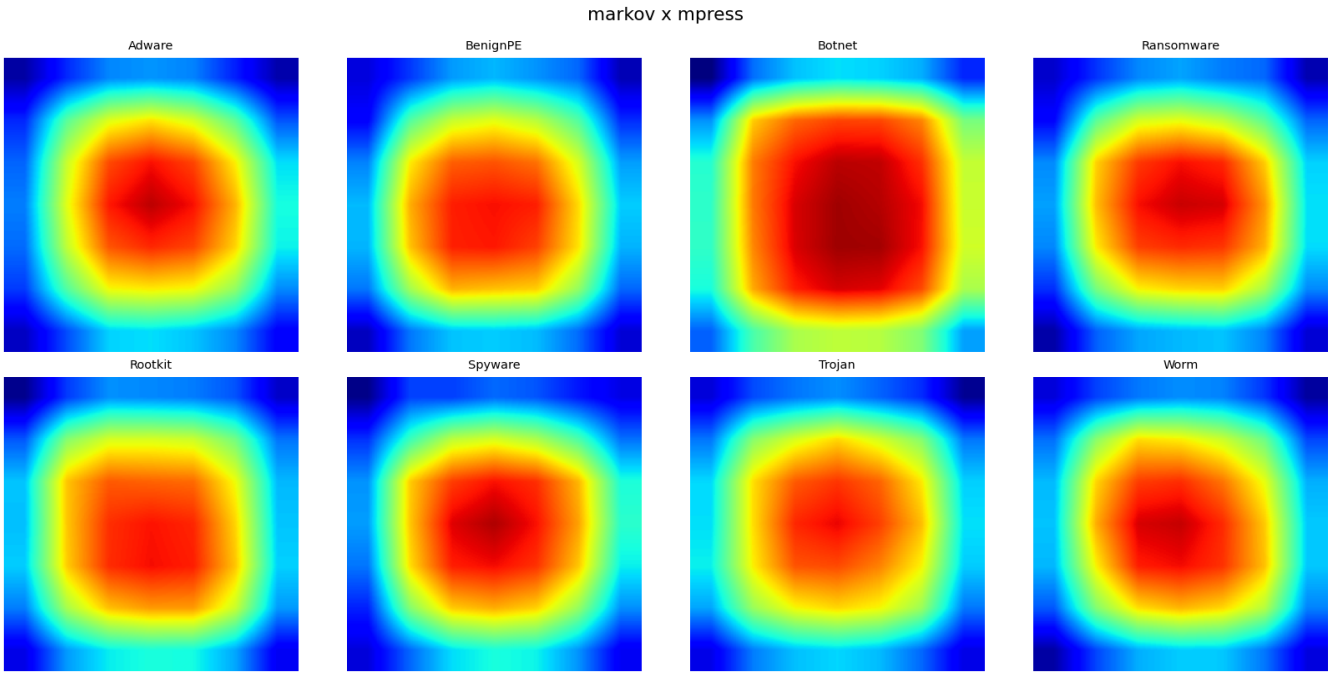


Figure 24: Cumulative class-level HiResCAM heatmaps for the markov encoding under the MPRESS packer, shown per family.

C.3 Entropy

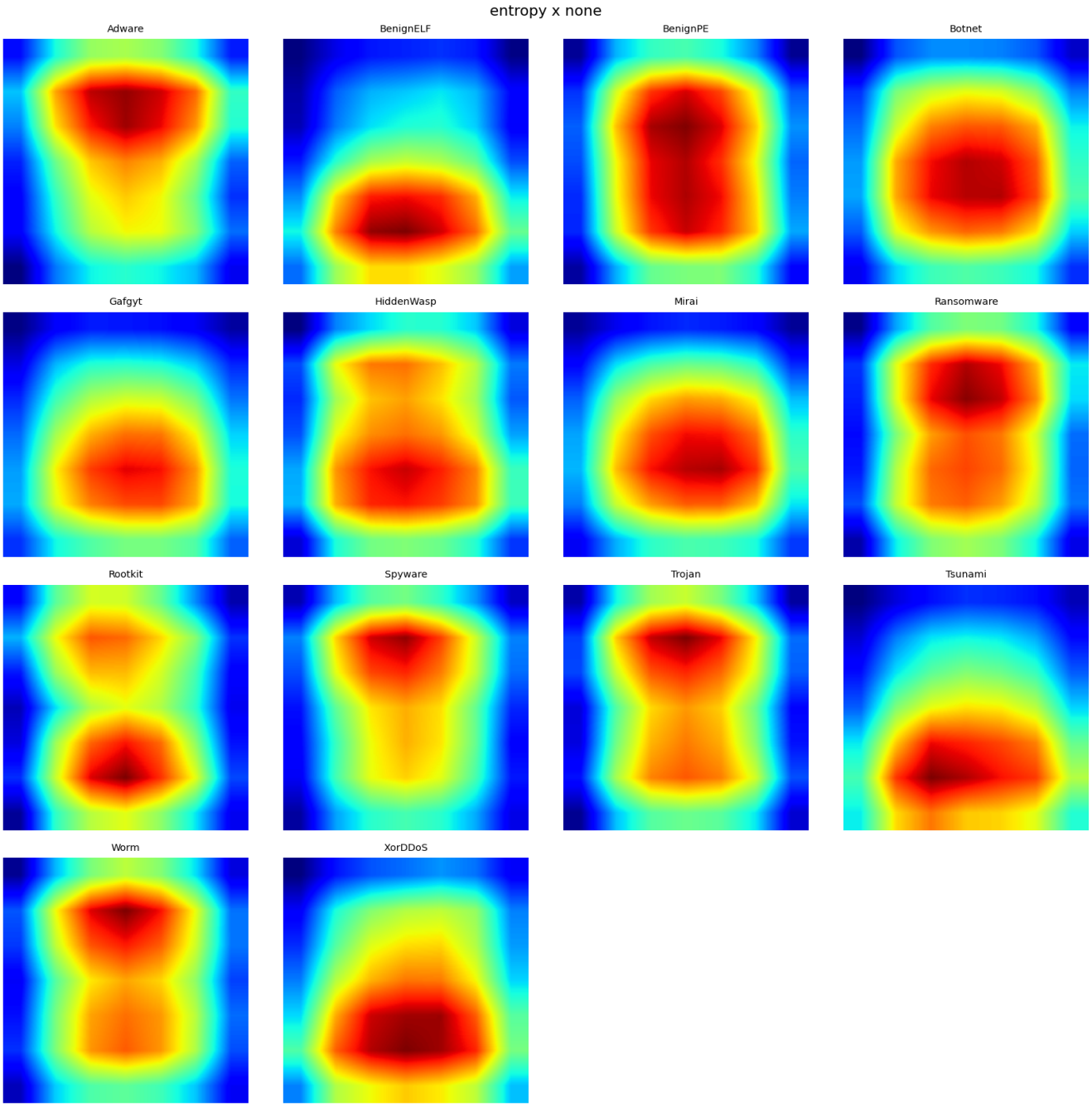


Figure 25: Cumulative class-level HiResCAM heatmaps for Entropy under the none packer, shown per family.

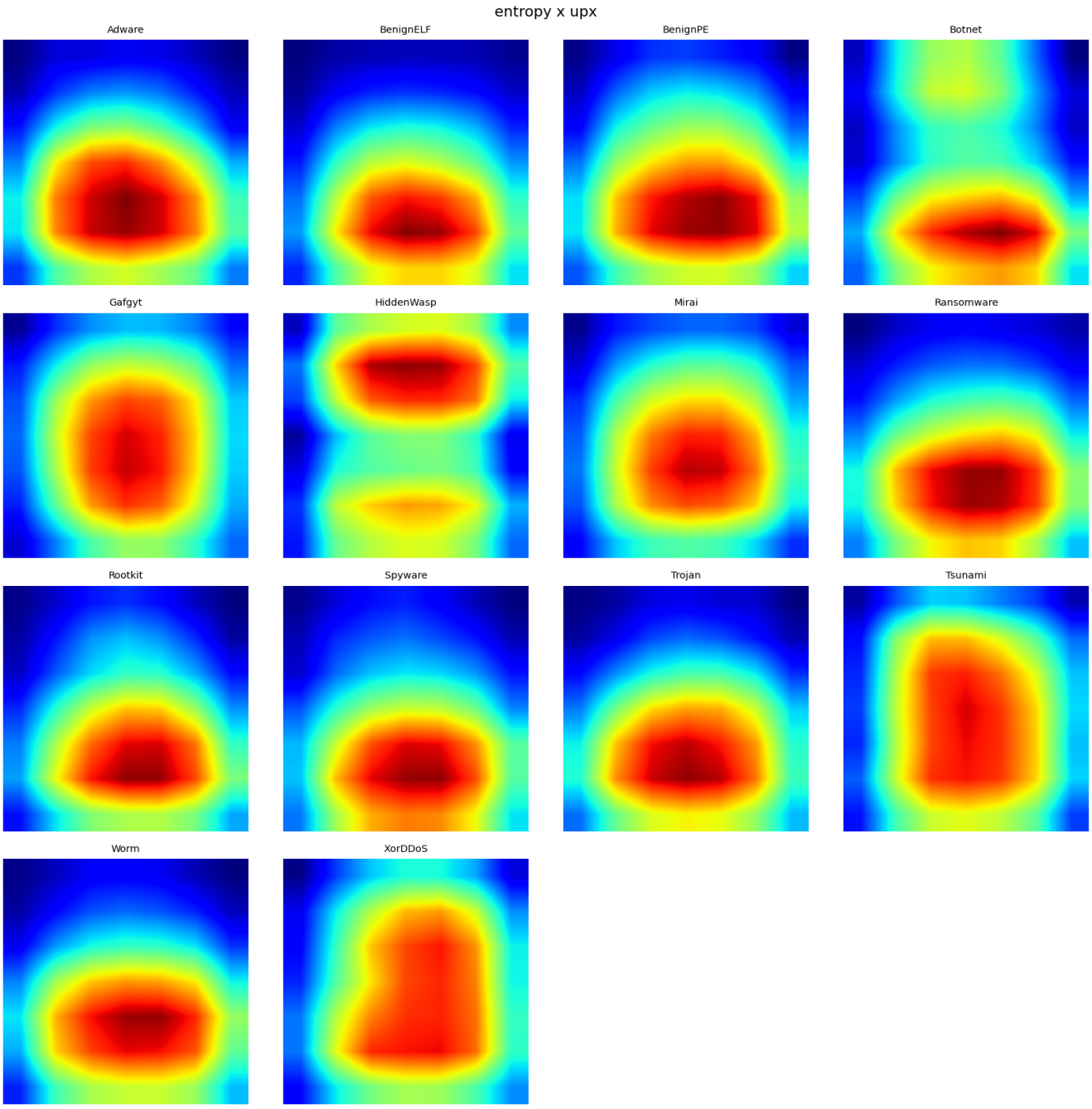


Figure 26: Cumulative class-level HiResCAM heatmaps for Entropy under the UPX packer, shown per family.

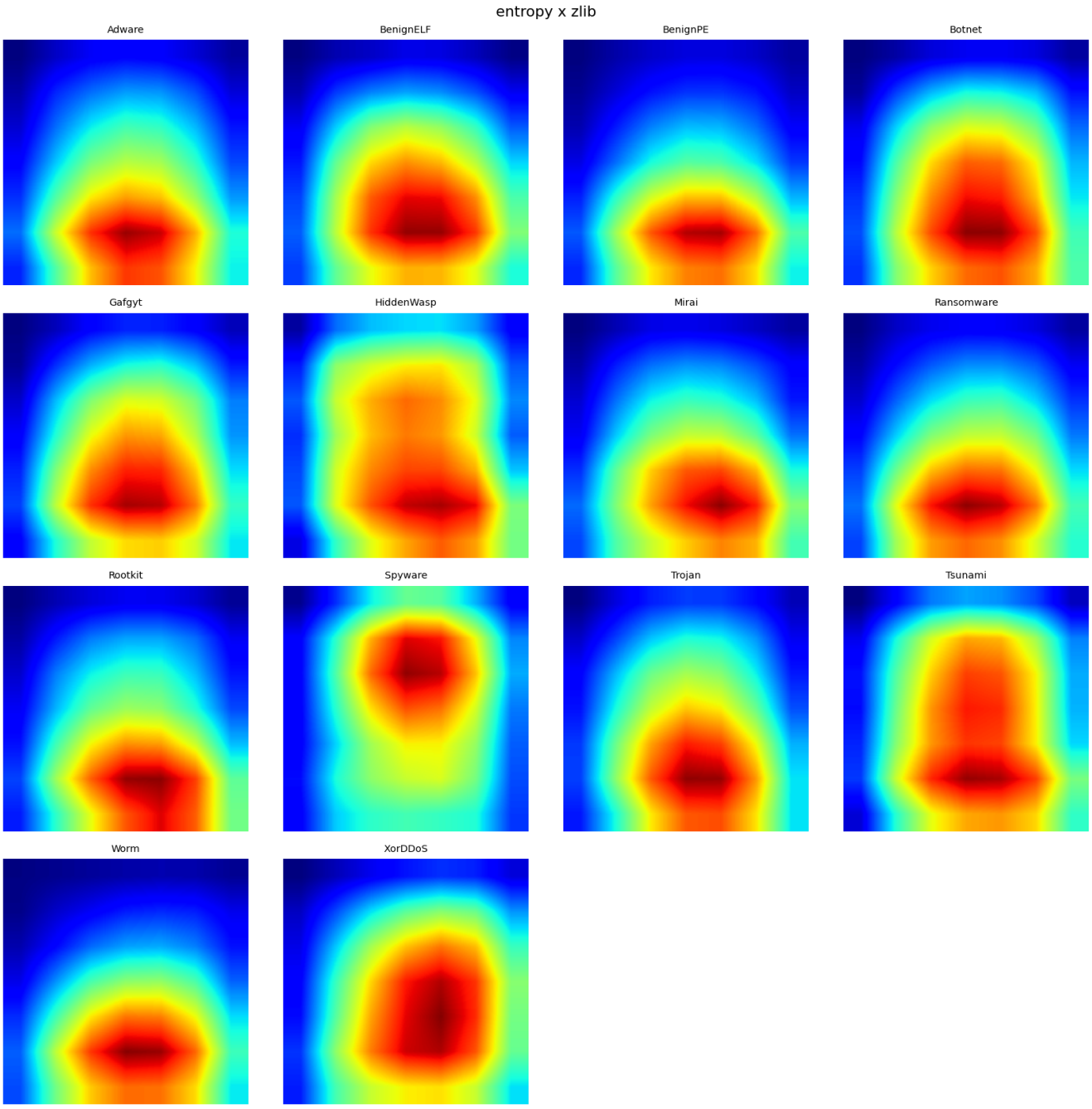


Figure 27: Cumulative class-level HiResCAM heatmaps for Entropy under the zlib packer, shown per family.

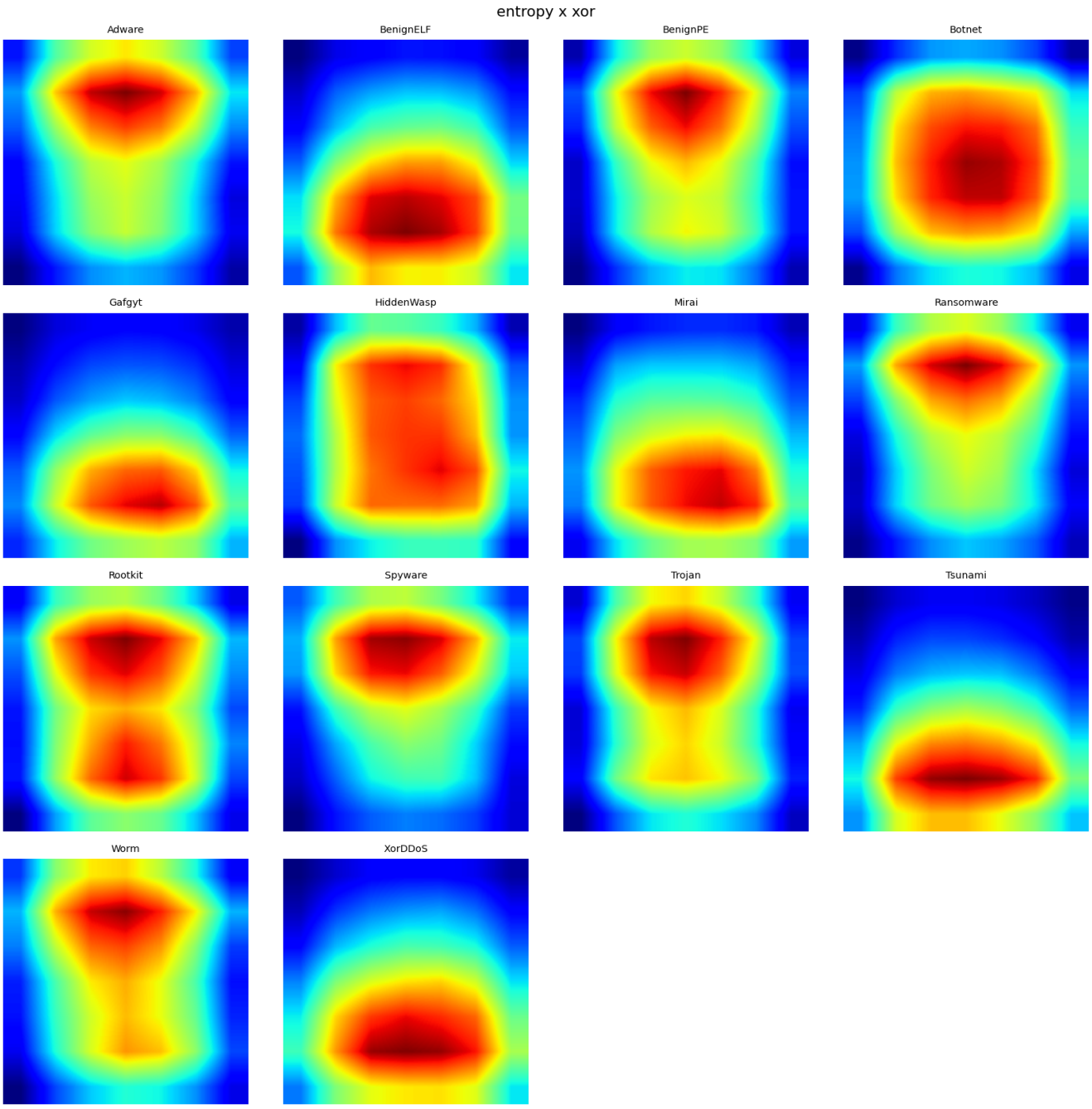


Figure 28: Cumulative class-level HiResCAM heatmaps for Entropy under the XOR packer, shown per family.

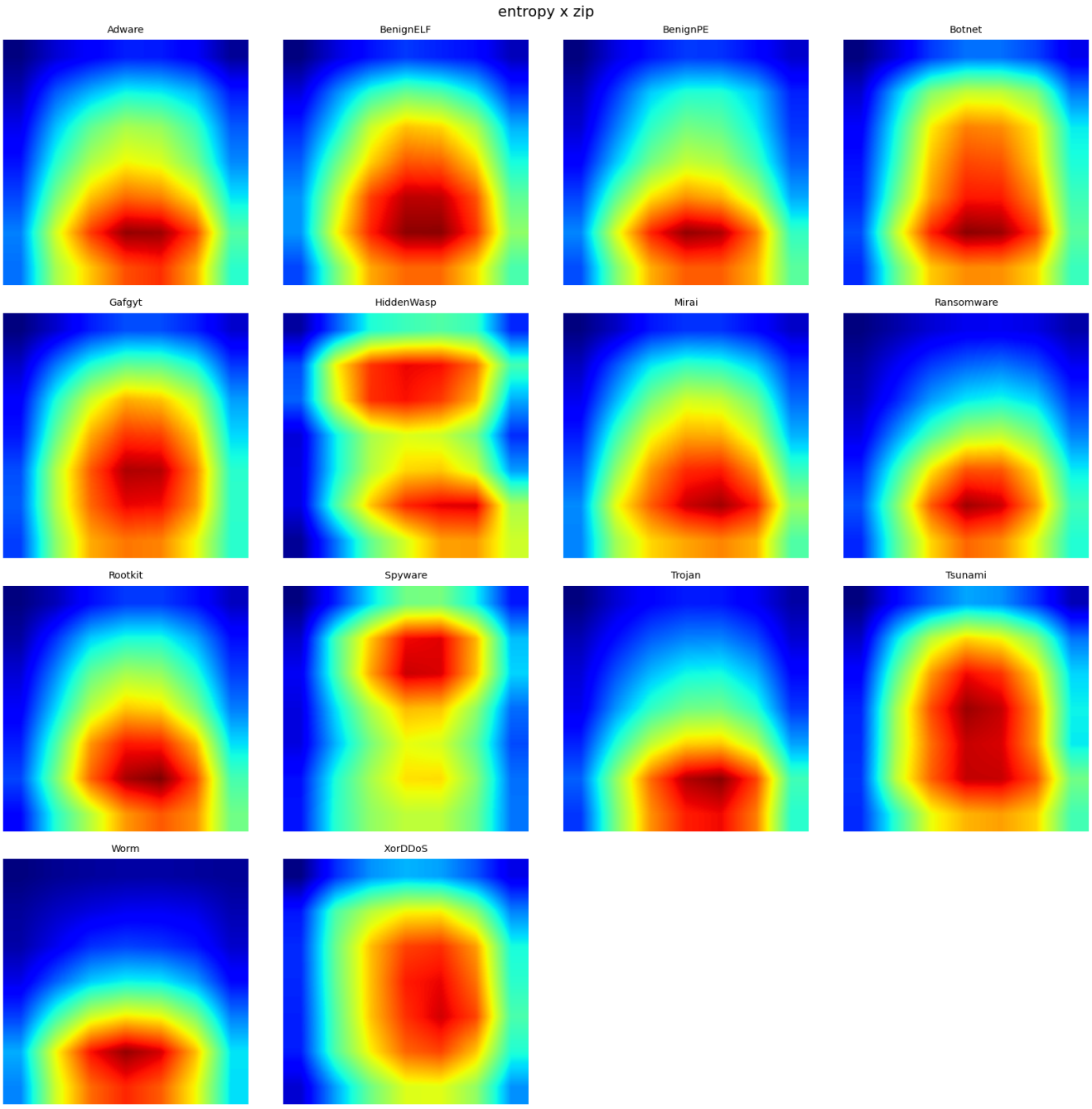


Figure 29: Cumulative class-level HiResCAM heatmaps for Entropy under the ZIP packer, shown per family.

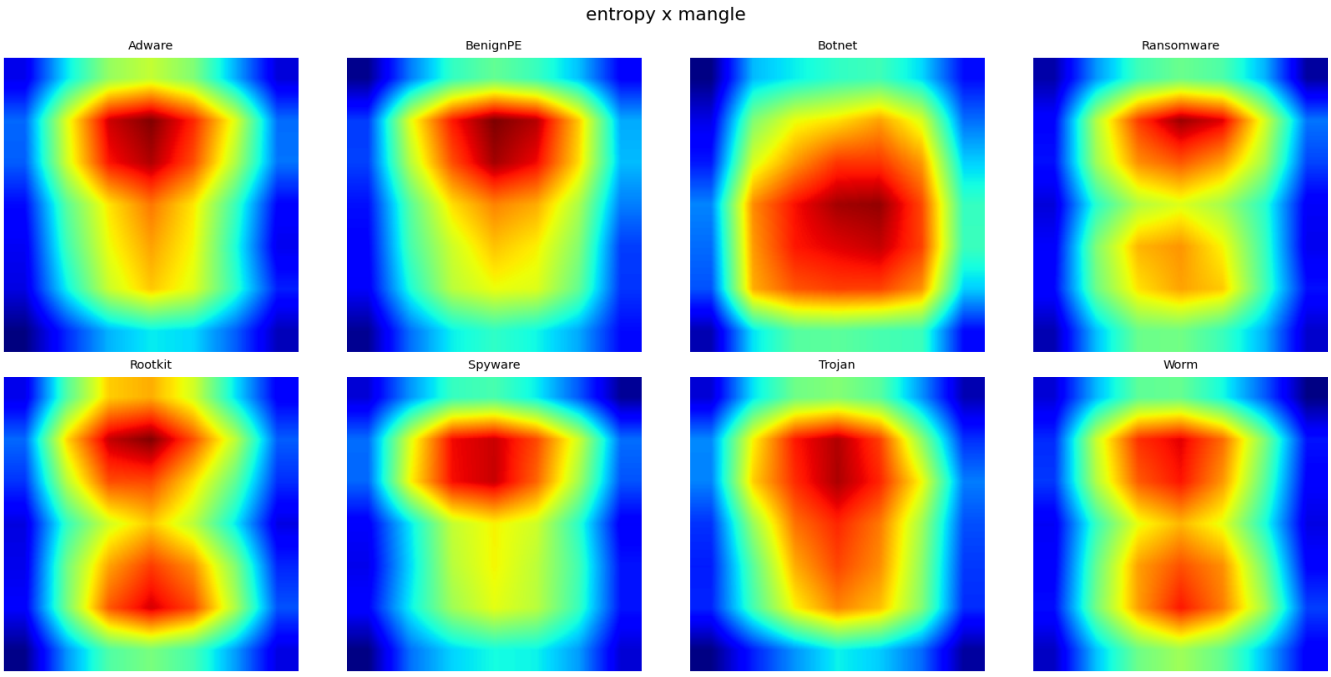


Figure 30: Cumulative class-level HiResCAM heatmaps for Entropy under the Mangle packer, shown per family.

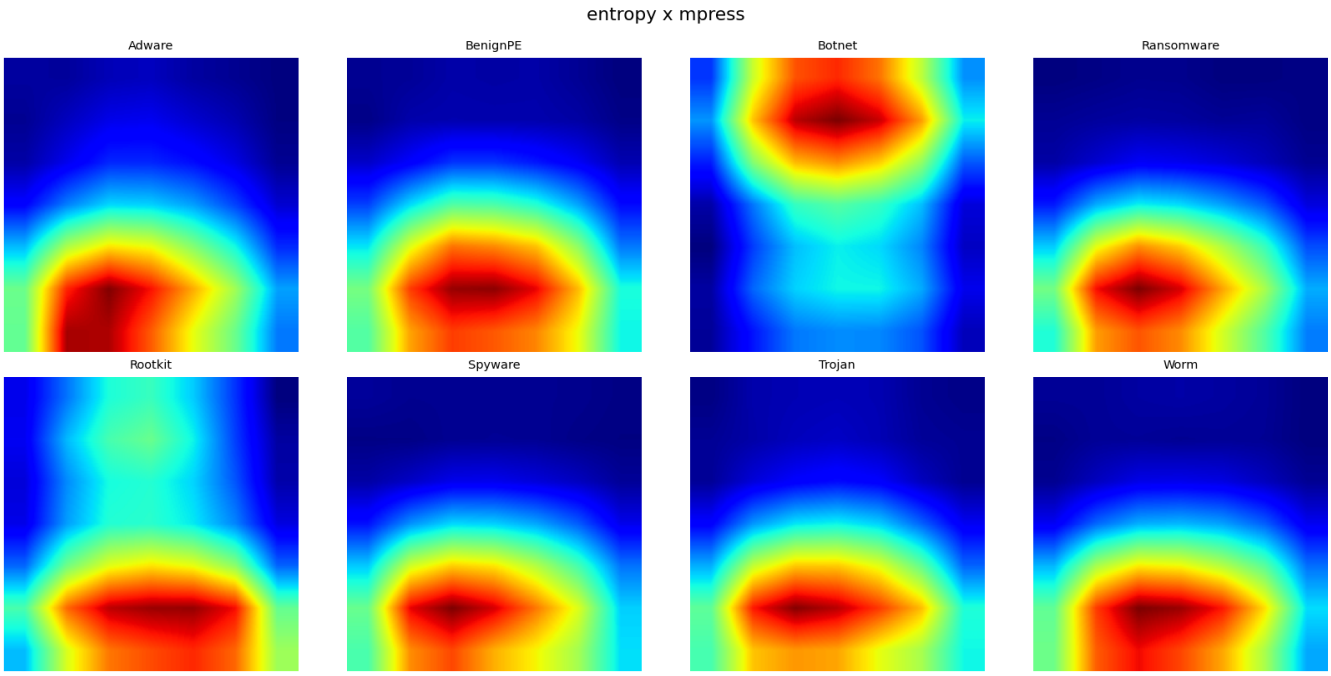


Figure 31: Cumulative class-level HiResCAM heatmaps for Entropy under the MPRESS packer, shown per family.