Malware Detection with CNNs on Entropy and Greyscale Images

ARTICLE HISTORY
Received 11 October 2025
Accepted 26 November 2025
Published 6 January 2026

Harry Darton
Sheffield Hallam University
School of Computing and Digital Technologies
Sheffield, United Kingdom
harryphuket@gmail.com
ORCID: 0009-0004-5674-2609

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026

H. Darton, "Malware Detection with CNNs on Entropy and Greyscale Images", Latin-American Journal of Computing (LAJC), vol. 13, no. 1, 2026.

Malware Detection with CNNs on Entropy and Greyscale Images

Harry Darton
Sheffield Hallam University
School of Computing and Digital Technologies
Sheffield, United Kingdom
harryphuket@gmail.com

***Abstract*— This study investigates whether convolutional neural networks (CNNs) trained on visual representations of Portable Executable (PE) files can rival traditional machine learning classifiers trained on engineered features. A dataset of over 200,000 PE files [1] was used to derive two feature sets (Basic and Ember-Lite) [2] and to generate 256x256 greyscale and entropy images [3],[4]. Three CNNs (SimpleCNN, ResNet-18 [5], EfficientNet-B0 [6]) were trained and evaluated against five baselines (Random Forest, XGBoost [7], CatBoost [8], LightGBM, Logistic Regression). Tree-based models with enriched features achieved the highest scores, with CatBoost reaching a ROC-AUC of 0.990. The best CNN, EfficientNet-B0 on entropy images, obtained a ROC-AUC of 0.954. Although CNNs did not surpass feature-based models, they showed competitive results when feature engineering was constrained. These findings indicate that visual approaches offer a promising alternative for static malware detection, particularly when combined with entropy-based representations [9].**

Keywords*— *malware detection, convolutional neural networks, entropy images, greyscale images, static analysis

I. Introduction

The rapid evolution of malicious software continues to pose a critical challenge to global cybersecurity. Traditional static and dynamic analysis techniques remain central to malware detection, yet their scalability and adaptability are increasingly strained by the volume and complexity of new samples emerging daily [10]. Static analysis, which inspects binary structure without execution, provides efficiency and safety but depends heavily on manually engineered features. These handcrafted representations are sensitive to obfuscation and require expert knowledge to maintain. Recent advances in deep learning have introduced new possibilities for automated feature extraction that may overcome such limitations [11].

Malware detection can be viewed as a binary classification problem in which a model must learn discriminative patterns between benign and malicious Portable Executable (PE) files. Earlier static approaches focused on syntactic features such as byte histograms, imported libraries, and section metadata [2]. While these features remain effective, they are often dataset-specific and may fail when the underlying malware distribution shifts. Convolutional Neural Networks (CNNs) offer an alternative pathway [11] by learning directly from raw or visually transformed data. In computer vision, CNNs have achieved outstanding success in recognizing complex spatial relationships within images. When applied to malware, they can automatically extract high-level spatial–statistical representations from a binary's byte sequence, potentially reducing reliance on expert feature engineering [12].

Transforming PE files into visual formats has gained attention because it allows direct use of image-based architectures without disassembling executables. Two representations have become prominent: greyscale images, formed by mapping byte values to pixel intensities, and entropy images [3],[4],[13], which highlight structural irregularities related to code density and compression. These visualizations reveal characteristic patterns that correspond to malware families, packing, and section entropy, providing CNNs with texture-like information unavailable to conventional static models. However, existing studies differ widely in dataset size, preprocessing, and evaluation methodology, making it difficult to assess [9],[14] whether CNN-based visual analysis can truly compete with established feature-based models.

Prior work has shown that tree-based ensembles such as Random Forest, XGBoost [7], CatBoost [8], and LightGBM deliver high accuracy on engineered feature sets like EMBER [2], often exceeding 0.99 ROC-AUC. Although several researchers have experimented with CNNs on image representations [3],[9],[15], many comparisons are indirect, rely on small datasets, or use pretrained vision networks without systematic control of variables. The literature therefore lacks a consistent large-scale benchmark that contrasts CNN performance with traditional models under identical data and evaluation conditions. Furthermore, while some studies report promising CNN results, few investigate how architectural complexity or input modality (greyscale vs entropy vs combined) affect performance [6],[15] or generalization.

This research addresses that gap through a controlled, large-scale comparison between visual-representation CNNs and feature-based classifiers for static malware detection. A dataset of more than 200,000 PE files was processed to generate both engineered features and 256x256 image representations [1]. Three CNN architectures of increasing depth, SimpleCNN, ResNet-18[5], and EfficientNet-B0 [6], were trained from scratch using greyscale, entropy, and dual-channel inputs. Their results were benchmarked against five tree-based baselines (Random Forest, XGBoost [7], CatBoost [8], LightGBM, and Logistic Regression) built on two feature sets: a compact Basic subset and an Ember-Lite extension. All models were evaluated under identical stratified splits and metrics, including ROC-AUC, F1-score, precision, recall, and accuracy.

The study contributes in three principal ways:

1. It provides a reproducible comparison between CNN-based and feature-based static detection using the same dataset and preprocessing pipeline.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: 10.5281/zenodo.17941249

2. It analyses the effect of image representation (greyscale, entropy, and combined) on CNN performance.
3. It evaluates how model depth influences the trade-off between feature learning capacity and overfitting risk.

By unifying these aspects, the work offers an empirical baseline for future research into vision-based malware detection and clarifies the extent to which CNNs can replace or complement engineered features [9].

II. Background and Related Work

Research on static malware detection has evolved from handcrafted feature extraction to representation-learning approaches based on neural networks. Early static pipelines focused on syntactic features derived from Portable Executable (PE) headers, import tables, and byte histograms [16], [17]. Ensemble methods such as Random Forest and XGBoost became widely adopted because they balanced predictive accuracy with interpretability.

Subsequent studies explored the visual encoding of binaries as two-dimensional matrices, where raw bytes were transformed into greyscale images to capture structural patterns [3]. This approach enabled convolutional networks to learn discriminative textures associated with packed or obfuscated code without explicit feature engineering. Han et al. [4] and Kalash et al. [12] extended this concept by employing deeper CNNs that achieved performance comparable to engineered-feature baselines.

Entropy-based representations introduced a further dimension by quantifying local randomness across the file, highlighting high-entropy regions characteristic of compression and encryption [15]. Brosolo et al. [14] demonstrated that combining byte and entropy channels improved separability between benign and malicious samples. However, most prior work relied on small datasets or inconsistent preprocessing, limiting reproducibility.

The present study addresses these limitations through a unified pipeline incorporating both feature-based and image-based approaches under controlled preprocessing and multi-seed evaluation. By comparing ensemble and CNN architectures directly on 200,000+ samples, it contributes empirical evidence on the relative merits of learned versus handcrafted representations for static malware detection.

III. Methodology

In this section, the methodology featured in Fig. 2 is explained as follows:

A. Dataset Construction

The experiments employed a static malware dataset containing more than 200,000 Portable Executable (PE) files, comprising roughly equal proportions of benign and malicious samples. Each file was validated to ensure accessibility and correct labelling. No dynamic execution or network traffic data were used, maintaining strict static conditions. A group-wise stratified split produced three subsets: 70% for training, 15% for validation, and 15% for independent testing. Stratification ensured proportional representation of malware families and preserved class balance across splits. All preprocessing, feature extraction, and model training were performed offline to eliminate any risk of infection.

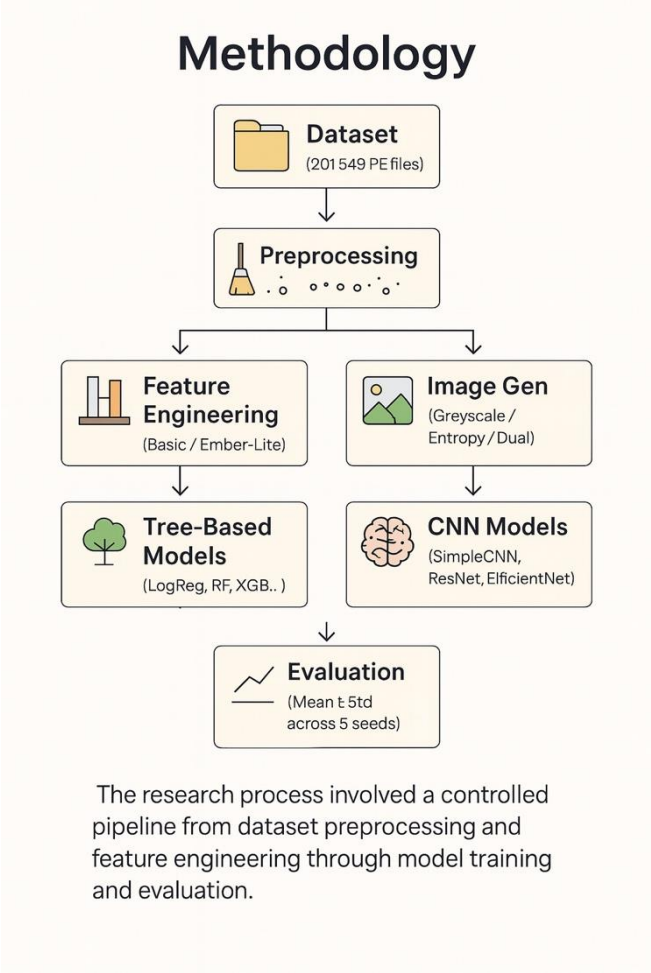

Fig. 2. Methodological workflow for dataset processing, model training, and evaluation.
The figure summarizes the experimental pipeline from dataset acquisition and preprocessing through feature engineering, image generation, and model evaluation across five random seeds.

B. Feature Engineering

Two engineered feature sets were created to provide traditional baselines.

1. Basic set: extracted lightweight structural attributes such as file size, section counts, imported library frequencies, and entropy statistics.
2. Ember-Lite set: extended the Basic features with a reduced subset of the EMBER 2018 dataset, including byte histograms, header metadata, and string features. The Ember-Lite feature set used here comprised only a small, computationally lightweight subset of the EMBER 2018 feature groups, selected to minimize pre-processing overhead whilst still producing a more detailed, EMBER-oriented set of features for comparison.

Both sets were scaled using min–max normalization. These vectors served as input to five machine-learning models: Random Forest, XGBoost, CatBoost, LightGBM, and Logistic Regression. Hyperparameters were tuned by grid search on the validation split.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: 10.5281/zenodo.17941249

C. Image Generation

For the visual-representation branch, each PE file was converted into two 256x256 images: a greyscale byte map and an entropy image. PE files shorter than 65,536 bytes were zero padded before reshaping and files exceeding this length were truncated so that all samples produced a consistent 256x256 matrix.

- The greyscale representation mapped each byte value (0-255) to a pixel intensity, preserving sequential order.
- The entropy representation applied a sliding-window Shannon entropy calculation to highlight structural irregularities related to code density, packing, and compression.

Images were stored as compressed PNGs and normalized to the [0, 1] range at load time. Three input modalities were tested: single-channel greyscale, single-channel entropy, and dual-channel combined images. The dual-channel version concatenated normalized greyscale and entropy matrices along the channel axis to retain spatial alignment. Representative examples of the three image modalities are shown in Fig. 1.

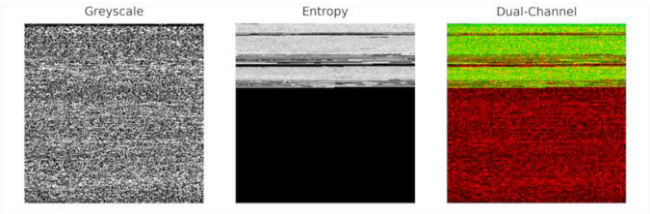

Fig. 1. Visualization of greyscale, entropy, and dual-channel image representations used for CNN training. The dual-channel view merges greyscale and entropy channels into red and green for clarity; CNNs process both as greyscale tensors.

D. CNN Architectures

Three convolutional neural networks of increasing depth and complexity were implemented to explore architectural effects.

1. SimpleCNN: a custom lightweight model with three convolutional blocks and max-pooling, designed to provide a minimal baseline.
2. ResNet-18: a residual architecture enabling deeper feature learning while mitigating vanishing-gradient issues.
3. EfficientNet-B0: a compound-scaled network optimizing depth, width, and resolution for parameter efficiency.

Each model concluded with a global average-pooling layer and a fully connected single sigmoid output neuron optimized with binary cross-entropy loss. All architectures were trained from scratch rather than fine-tuned from natural-image weights to maintain domain specificity.

More recent vision architectures exist, but the chosen trio provides a controlled progression from shallow to moderately deep networks, enabling a fair comparison of representational capacity while keeping reproducibility and computational cost practical for large-scale malware datasets.

E. Training Procedure

Training was performed using the PyTorch 2.8 framework on GPU hardware. Batch size, learning rate, and weight decay were tuned empirically through pilot runs. Early stopping based on validation loss prevented overfitting, and the best model weights were checkpointed. Feature vectors and image tensors were normalized to the [0, 1] range prior to training. Early experiments confirmed that standardization to [-1, 1] or z-scoring did not improve convergence for models trained from scratch. All random seeds, splits, and preprocessing parameters were fixed for reproducibility. During training, data augmentation was limited to horizontal and vertical flips to avoid distorting binary layout information. Each experiment was repeated across five random seeds to estimate variability.

F. Evaluation Metrics

Model performance was assessed on the held-out test set using multiple metrics:

- ROC-AUC as the principal discrimination measure.
- F1-score, precision, recall, and accuracy for complementary evaluation.
- Calibration curves and confusion matrices for selected runs to examine reliability and error distribution.

All results were reported as mean ± standard deviation across seeds. Timing and resource usage were recorded but not compared formally because training occurred on heterogeneous hardware.

IV. Deployment And Implementation

All experiments were executed on a Windows 11 workstation equipped with an NVIDIA RTX 5080 GPU (16 GB VRAM), a Ryzen 7 7800X3D CPU, and 32 GB RAM. The environment used Python 3.12 and PyTorch 2.8.0 (CUDA 12.8) within a PyCharm-managed virtual environment. Dataset preprocessing and feature extraction were performed offline to prevent malware execution risk.

A. Dataset and Preprocessing

A total of 201,549 PE files from Lester (2021) were processed into multiple representations: (i) Basic PE features, (ii) extended Ember-Lite features, (iii) greyscale byte-maps, (iv) entropy images computed over 32-byte windows, and (v) dual-channel stacks combining greyscale and entropy tensors. Group-wise stratification by Imphash ensured partition integrity across train, validation, and test splits, mitigating leakage.

B. Model Implementation

Tree-based models (Logistic Regression, Random Forest, XGBoost, LightGBM, CatBoost) were implemented via Scikit-learn and native library APIs using identical folds and hyperparameter grids. For CNNs, three architectures were selected to represent increasing structural depth: SimpleCNN (three convolutional layers), ResNet-18, and EfficientNet-B0. All networks employed batch normalization, ReLU activations, early stopping, and Adam optimization with learning-rate scheduling.

C. Training Protocol

Training used Adam (learning rate = $1x10^{-3}$) with CrossEntropyLoss, ReduceLROnPlateau (patience = 2), early stopping (patience = 5), and batch size 128, capped at 40

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: 10.5281/zenodo.17941249

epochs. Only the best-performing checkpoint (lowest validation loss) per run was retained for test evaluation. Results were averaged across seeds (42–46) and reported with standard deviation.

D. Reproducibility Controls

All outputs (model weights, logs, and metrics.json files) were stored under versioned directories for full traceability. Reproducibility was validated by selectively rerunning a subset of experiments on different hardware (MX250 laptop and MacBook CPU), confirming consistent metrics within variance bounds. The entire pipeline is available via a public GitHub repository.

V. Analysis and Discussion of Findings

A. Overview

The study provides a comprehensive comparative analysis, combining large-scale data, multiple feature representations, and a multi-seed evaluation to support robust and generalizable conclusions.

This section presents the quantitative results obtained from both the feature-based and CNN-based branches of the study. All models were evaluated on the held-out test partition using identical metrics and configuration to ensure comparability. Performance values reported correspond to the mean of five random-seed runs, with standard deviation shown where applicable.

B. Feature-Based Baselines

Traditional classifiers trained on the Basic and Ember-Lite feature sets produced strong results across all metrics. The Basic feature set already provided a solid baseline, achieving ROC-AUC scores above 0.96 for ensemble methods. Incorporating the additional Ember-Lite attributes further improved separability between benign and malicious samples.

Among the evaluated models, CatBoost delivered the highest and most consistent performance, reaching 0.990 ± 0.001 ROC-AUC and an F1-score of 0.947 ± 0.005. XGBoost and LightGBM followed closely, differing by less than 0.002 ROC-AUC, while Random Forest achieved comparable accuracy with slightly higher variance. Logistic Regression, as expected, under-fit the nonlinear relationships and produced the lowest AUC (≈ 0.94). These outcomes confirm that gradient-boosted tree ensembles remain a robust benchmark for static malware detection when high-quality engineered features are available.

C. CNN Performance on Greyscale and Entropy Images

The CNN experiments evaluated three architectures of increasing complexity (SimpleCNN, ResNet-18, and EfficientNet-B0) across three input modalities: greyscale, entropy, and dual-channel combined images.

- Greyscale Images: captured structural layout but limited semantic variation. SimpleCNN achieved 0.886 ± 0.011 ROC-AUC, ResNet-18 0.937 ± 0.008, and EfficientNet-B0 0.931 ± 0.010.
- Entropy Images: provided higher discriminative information due to encoding of randomness and packing density. Here, EfficientNet-B0 achieved 0.954 ± 0.005 and 0.898 ± 0.008 F1, the best CNN result overall, indicating that entropy-based spatial cues are particularly informative for distinguishing obfuscated malware.
- Combined Images: merging greyscale and entropy channels offered only marginal gains for shallower networks and, in some cases, introduced redundancy. ResNet-18 reached 0.950 ± 0.009 ROC-AUC, while the dual-channel variant of EfficientNet-B0 achieved 0.947 ± 0.004, showing little additional benefit.

Training variability across seeds remained low (±0.002 AUC), confirming stable convergence. Validation loss curves indicated earlier saturation for SimpleCNN, while deeper models continued improving over more epochs, reflecting greater representational capacity.

D. Comparative Analysis

A direct comparison between the best feature-based and CNN models highlights a clear but narrowing performance gap. CatBoost (Ember-Lite) exceeded EfficientNet-B0 (Entropy) by approximately 0.036 ROC-AUC, achieved 0.983 ± 0.001 precision and 0.913 ± 0.011 recall compared with 0.954 ± 0.004 precision and 0.848 ± 0.013 recall for the CNN. When feature engineering is limited, CNNs trained on entropy visualizations approach ensemble-level accuracy without any handcrafted inputs.

Fig. 3 visualizes the overall comparison. Feature-based ensembles dominate the upper bound, while CNNs occupy a competitive mid-band with notably reduced preprocessing overhead.

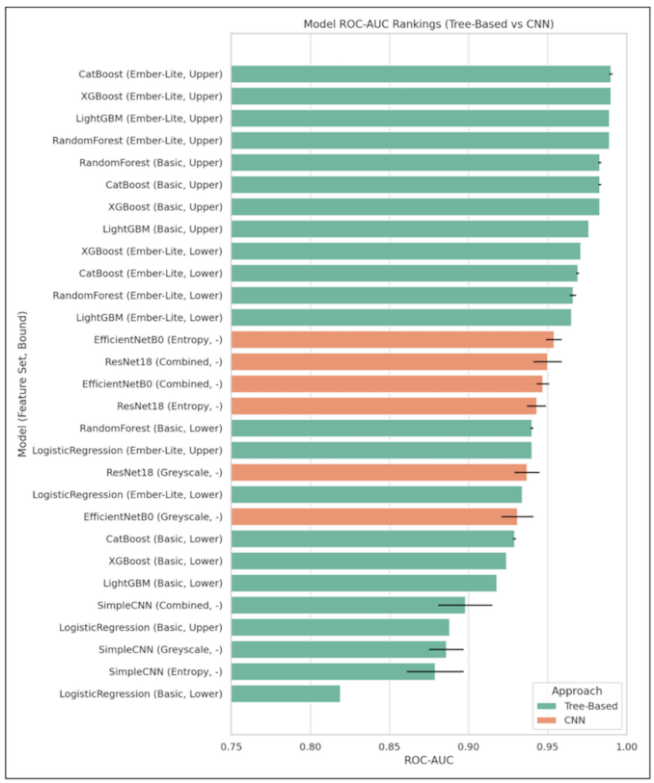

Fig. 3. Overall performance comparison of tree-based and CNN-based models across all input modalities

Table I reports mean ± standard deviation over five random-seed runs for all models across Basic, Ember-Lite, greyscale, entropy, and combined inputs.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: 10.5281/zenodo.17941249

TABLE I. MODEL PERFORMANCE COMPARISON (FEATURE-BASED VS CNN)
Summary of quantitative performance metrics (mean ± standard deviation) for all models across Basic, Ember-Lite, Greyscale, Entropy, and Combined input representations.

Model	Feature Set	Bound	ROC-AUC	F1	Accuracy	Precision	Recall
CatBoost	Ember-Lite	Upper	0.990 ± 0.001	0.947 ± 0.005	0.941 ± 0.005	0.983 ± 0.001	0.913 ± 0.011
XGBoost	Ember-Lite	Upper	0.990 ± 0.000	0.940 ± 0.000	0.933 ± 0.000	0.983 ± 0.000	0.900 ± 0.000
LightGBM	Ember-Lite	Upper	0.989 ± 0.000	0.936 ± 0.000	0.929 ± 0.000	0.982 ± 0.000	0.894 ± 0.000
RandomForest	Ember-Lite	Upper	0.989 ± 0.000	0.929 ± 0.004	0.922 ± 0.004	0.977 ± 0.003	0.886 ± 0.008
RandomForest	Basic	Upper	0.983 ± 0.001	0.932 ± 0.003	0.924 ± 0.003	0.966 ± 0.007	0.900 ± 0.010
CatBoost	Basic	Upper	0.983 ± 0.001	0.938 ± 0.002	0.930 ± 0.002	0.960 ± 0.003	0.916 ± 0.005
XGBoost	Basic	Upper	0.983 ± 0.000	0.937 ± 0.000	0.929 ± 0.000	0.963 ± 0.000	0.912 ± 0.000
LightGBM	Basic	Upper	0.976 ± 0.000	0.919 ± 0.000	0.911 ± 0.000	0.959 ± 0.000	0.882 ± 0.000
XGBoost	Ember-Lite	Lower	0.971 ± 0.000	0.898 ± 0.000	0.891 ± 0.000	0.980 ± 0.000	0.829 ± 0.000
CatBoost	Ember-Lite	Lower	0.969 ± 0.001	0.894 ± 0.005	0.887 ± 0.005	0.979 ± 0.003	0.822 ± 0.009
RandomForest	Ember-Lite	Lower	0.966 ± 0.002	0.904 ± 0.005	0.897 ± 0.005	0.976 ± 0.001	0.842 ± 0.008
LightGBM	Ember-Lite	Lower	0.965 ± 0.000	0.902 ± 0.000	0.895 ± 0.000	0.971 ± 0.000	0.843 ± 0.000
EfficientNetB0	Entropy	-	0.954 ± 0.005	0.898 ± 0.008	0.889 ± 0.008	0.954 ± 0.004	0.848 ± 0.013
ResNet18	Combined	-	0.950 ± 0.009	0.904 ± 0.010	0.890 ± 0.015	0.916 ± 0.044	0.895 ± 0.039
EfficientNetB0	Combined	-	0.947 ± 0.004	0.898 ± 0.006	0.888 ± 0.006	0.949 ± 0.005	0.852 ± 0.011
ResNet18	Entropy	-	0.943 ± 0.006	0.897 ± 0.008	0.888 ± 0.008	0.954 ± 0.014	0.847 ± 0.014
RandomForest	Basic	Lower	0.940 ± 0.001	0.879 ± 0.002	0.868 ± 0.002	0.930 ± 0.001	0.833 ± 0.003
LogisticRegression	Ember-Lite	Upper	0.940 ± 0.000	0.855 ± 0.000	0.844 ± 0.000	0.927 ± 0.000	0.793 ± 0.000
ResNet18	Greyscale	-	0.937 ± 0.008	0.862 ± 0.013	0.854 ± 0.012	0.943 ± 0.006	0.795 ± 0.025
LogisticRegression	Ember-Lite	Lower	0.934 ± 0.000	0.876 ± 0.000	0.863 ± 0.000	0.917 ± 0.000	0.838 ± 0.000
EfficientNetB0	Greyscale	-	0.931 ± 0.010	0.880 ± 0.008	0.871 ± 0.006	0.945 ± 0.017	0.824 ± 0.027
CatBoost	Basic	Lower	0.929 ± 0.001	0.874 ± 0.002	0.862 ± 0.003	0.927 ± 0.005	0.826 ± 0.002
XGBoost	Basic	Lower	0.924 ± 0.000	0.874 ± 0.000	0.862 ± 0.000	0.918 ± 0.000	0.835 ± 0.000
LightGBM	Basic	Lower	0.918 ± 0.000	0.872 ± 0.000	0.860 ± 0.000	0.918 ± 0.000	0.831 ± 0.000
SimpleCNN	Combined	-	0.898 ± 0.017	0.777 ± 0.153	0.788 ± 0.095	0.898 ± 0.055	0.728 ± 0.210
LogisticRegression	Basic	Upper	0.888 ± 0.000	0.801 ± 0.000	0.784 ± 0.000	0.853 ± 0.000	0.755 ± 0.000
SimpleCNN	Greyscale	-	0.886 ± 0.011	0.849 ± 0.011	0.833 ± 0.017	0.886 ± 0.035	0.817 ± 0.015
SimpleCNN	Entropy	-	0.879 ± 0.018	0.843 ± 0.012	0.835 ± 0.009	0.931 ± 0.020	0.771 ± 0.032
LogisticRegression	Basic	Lower	0.819 ± 0.000	0.757 ± 0.000	0.733 ± 0.000	0.798 ± 0.000	0.720 ± 0.000

To further illustrate error distribution between benign and malicious classifications, Fig. 4 presents normalized confusion matrices for the best-performing ensemble (CatBoost) and CNN (EfficientNet-B0) models.

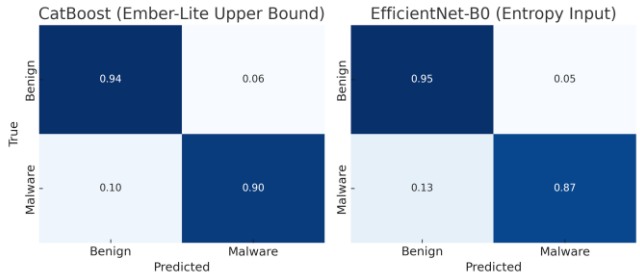

Fig. 4. Normalized confusion matrices comparing the best ensemble (CatBoost – Ember-Lite Upper Bound) and CNN (EfficientNet-B0 – Entropy Input) models. CatBoost achieves slightly superior precision on benign samples, while EfficientNet-B0 maintains strong recall on malware detection, demonstrating the narrowing gap between traditional and visual-based static analysis approaches.

Examination of the confusion matrices showed that CNNs occasionally misclassified small or lightly obfuscated malware samples, particularly where entropy images remained sparse after padding. In contrast, the tree-based models sometimes mislabeled benign files exhibiting elevated entropy or atypical section structures. These patterns are consistent with the feature sensitivities of each model family and help explain why tree-based models still retain a small overall advantage.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026
DOI: 10.5281/zenodo.17941249

E. Combined Image Representation and Interpretation

The hypothesis that merging greyscale and entropy inputs would enhance classification performance was not supported by the results. Across all CNN architectures, the dual-channel variant produced near-identical or slightly inferior ROC-AUC values compared with single-channel entropy inputs. This outcome reflects an important characteristic of PE visualization: while greyscale mappings and entropy images appear visually distinct, they encode strongly correlated structural information.

The greyscale representation captures the sequential byte distribution of sections, making dense code regions appear darker and sparse or zero-padded regions lighter. Entropy visualization, computed over fixed 32-byte windows, highlights the same structural boundaries by assigning higher intensity to compressed or encrypted blocks and lower intensity to static resources or padding. When concatenated, both modalities effectively describe the same transitions in spatial density and randomness. This redundancy dilutes gradient salience during training, as filters in early CNN layers receive conflicting but overlapping cues, hindering convergence to strongly discriminative features.

From a signal-processing perspective, direct stacking of channels also constrains the network to treat the two modalities as spatially aligned, which may not reflect semantic complementarity. A more effective fusion could involve learned attention or adaptive weighting between channels, allowing the model to emphasize entropy cues where they are most informative. Another avenue would be RGB-style compositing, in which greyscale, entropy, and a derived statistical feature (such as local variance or byte-frequency gradient) are encoded into separate color channels. This approach may exploit richer feature interactions akin to texture analysis in natural images. Alternatively, late fusion strategies (where embeddings from separate CNN branches are combined at the dense layer stage) could preserve each modality's distinct representational space while capturing joint correlations.

The observed stagnation of performance in combined inputs therefore does not imply redundancy between visual and entropy domains per se, but rather that naive spatial concatenation fails to capitalize on their complementary nature. Future research should explore cross-channel attention, feature-level fusion, or transformer-based architectures capable of learning non-linear relationships between modality-specific embeddings.

F. Interpretation of Key Findings

Three principal insights emerge from the experimental results.

1. Entropy images outperform greyscale inputs, revealing high-entropy packing and obfuscation regions strongly correlated with malicious binaries. The results indicate that entropy-based CNNs capture discriminative information unavailable from structural byte layouts alone.
2. Model depth directly influences performance. Deeper CNNs such as ResNet-18 and EfficientNet-B0 consistently surpass SimpleCNN, confirming that hierarchical feature extraction enhances visual malware representation learning.
3. Feature engineering versus representation learning: feature-based ensembles remain superior when rich handcrafted features are available, but CNNs offer a promising alternative for scalable, fully automated pipelines where feature computation is infeasible or costly.

The comparative results highlight that entropy visualizations preserve generalizable statistical cues about randomness and code density, while feature-engineered models exploit explicit semantic attributes. The best CNN configuration (EfficientNet-B0 trained on entropy images) achieved 0.954 ± 0.005 ROC-AUC, compared with 0.990 ± 0.001 for the CatBoost ensemble on the Ember-Lite feature set. This ~ 0.036 AUC difference demonstrates a narrowing gap between learned visual and engineered feature representations, even though ensemble methods remain the upper bound.

Beyond comparative accuracy, these results highlight a wider methodological shift in static malware research. As handcrafted features reach maturity, incremental performance gains often come at the cost of increased preprocessing complexity and domain dependence. Visual learning approaches, by contrast, demonstrate the capacity to generalize across unseen binaries with minimal feature engineering. Their scalability and model-agnostic input pipeline make them well suited for future integration into hybrid detection frameworks, combining the interpretability of engineered features with the adaptability of deep representation learning.

G. Broader Implications and Limitations

Although extensive, this comparison remains limited to static byte-level analysis and does not incorporate dynamic or hybrid behavioral features. Training and evaluation were conducted on heterogeneous hardware, and while reproducibility was confirmed, computational cost was not measured formally. Furthermore, dataset balance was maintained artificially, whereas real-world malware corpora are often heavily skewed.

These constraints suggest several research extensions: combining static and dynamic modalities; benchmarking under naturally imbalanced distributions; and assessing architecture efficiency across larger datasets or through transfer learning.

The dataset contained 201,549 executables, comprising 86,812 benign files and 114,737 malware samples. Although this distribution is only moderately skewed towards malicious files, real-world environments are typically dominated by benign software. Future evaluations should therefore assess performance and calibration under more strongly imbalanced conditions, or more benign-dominated conditions.

Nevertheless, the findings reinforce the viability of CNN-based static detectors as interpretable, automated complements to feature-based models. With entropy-derived visual inputs, CNNs achieved competitive performance within 3-4% ROC-AUC of state-of-the-art ensembles while eliminating manual feature design, indicating strong potential for scalable, real-time malware triage systems.

VI. Conclusion

This study compared feature-based machine-learning models and convolutional neural networks (CNNs) for static malware detection using a dataset of more than 200,000 Portable Executable files [1]. The experimental design provided a unified benchmark in which both traditional classifiers and visual deep-learning models were trained and evaluated under identical conditions.

The results demonstrated that tree-based ensembles, particularly CatBoost [8], remain the most accurate static detectors when high-quality handcrafted features are available, achieving 0.990 ± 0.001 ROC-AUC and 0.947 ± 0.005 F1. However, CNNs trained directly on byte-derived image representations achieved competitive performance without manual feature engineering. Among the visual models, entropy-based inputs consistently outperformed greyscale and combined modalities, and deeper networks such as ResNet-18 [5] and EfficientNet-B0 [6] significantly exceeded the accuracy of the shallow SimpleCNN. These findings confirm that entropy visualization provides strong discriminative cues and that model depth enhances representational capacity in image-based malware analysis [3],[4].

The comparative analysis revealed a narrowing gap of approximately 0.036 ROC-AUC between the best feature-based and CNN models, suggesting that vision-driven approaches can serve as scalable, automated alternatives when handcrafted features are unavailable or costly to compute.

Future work should explore hybrid static–dynamic pipelines that combine image-based deep learning with lightweight engineered features to further improve robustness against obfuscation and dataset drift [14]. Benchmarking computational efficiency across uniform hardware and assessing real-world, imbalanced data distributions would also strengthen the practical applicability of visual malware detection methods.

The limited improvement from the combined greyscale–entropy representation further highlights the challenge of naive multimodal fusion. Although the two inputs differ visually, their spatially correlated content may reduce discriminative gradients when processed jointly; suggesting that future architectures should employ attention-based fusion or late-stage embedding integration to better exploit complementary features without introducing redundancy.

Funding Statement

This research received no external funding. All computational work and analysis were conducted using personal and institutional resources without third-party support.

Acknowledgment

The author thanks Dr. Shahrzad Zargari of Sheffield Hallam University for supervision and constructive feedback throughout the research project.

Author Contributions

Harry Darton: Conceptualization, Methodology, Data Curation, Formal Analysis, Writing – Original Draft, Writing – Review & Editing.

References

[1] M. Lester, “PE malware machine learning dataset [Data set],” Practical Security Analytics, 2021. [Online]. Available: https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/

[2] H. S. Anderson and P. Roth, “EMBER: An open dataset for training static PE malware machine learning models,” arXiv preprint, 2018. [Online]. Available: https://arxiv.org/abs/1804.04637

[3] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in Proc. 8th Int. Symp. Visualization for Cyber Security (VizSec 2011), pp. 1–7, ACM, 2011. doi: 10.1145/2016904.2016908

[4] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, “Malware analysis using visualized images and entropy graphs,” Int. J. Inf. Security, vol. 14, no. 1, p. 1, 2014. doi: 10.1007/s10207-014-0242-0

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2016), pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90

[6] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. 36th Int. Conf. Machine Learning (ICML 2019), vol. 97, pp. 6105–6114, PMLR, 2019. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[7] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD ’16), pp. 785–794, ACM, 2016. doi: 10.1145/2939672.2939785

[8] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “CatBoost: Unbiased boosting with categorical features,” in Proc. 32nd Int. Conf. Neural Information Processing Systems (NeurIPS 2018), pp. 6639–6649, 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

[9] A. Bensaoud, N. Abudawaood, and J. Kalita, “Classifying malware images with convolutional neural network models,” arXiv preprint, 2020. [Online]. Available: https://arxiv.org/abs/2010.16108

[10] AV-TEST Institute, “Malware statistics & trends report,” 2024. [Online]. Available: https://www.av-test.org/en/statistics/malware/

[11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539

[12] M. Kalash et al., “Malware classification with deep convolutional neural networks,” in Proc. 10th Int. Conf. New Technologies, Mobility and Security (NTMS), pp. 1–5, IEEE, 2018. doi: 10.1109/NTMS.2018.8328749

[13] C. E. Shannon, “A mathematical theory of communication,” Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x

[14] M. Brosolo and M. Conti, “The road less travelled: Investigating robustness and explainability in CNN malware detection,” arXiv preprint, 2025. doi: 10.48550/arXiv.2503.01391

[15] B. Al-Masri, N. Bakir, A. El-Zaart, and K. Samrouth, “Dual convolutional malware network (DCMN): An image-based malware classification using dual convolutional neural networks,” Electronics, vol. 13, no. 18, p. 3607, 2024. doi: 10.3390/electronics13183607

[16] J. Saxe and K. Berlin, “Deep neural network based malware detection using two-dimensional binary program features,” arXiv preprint arXiv:1508.03096, 2015.

[17] E. Raff et al., “Malware detection by eating a whole EXE,” arXiv preprint arXiv:1710.09435, 2017.

Note on the Use of Artificial Intelligence (AI):

AI tools were used only for minor language editing and reference formatting. All methodological design, analysis, data interpretation, and writing decisions were performed by the author.

AUTHORS

Harry Darton

Harry Darton graduated from Sheffield Hallam University with a first class honours degree in Cyber Security and Digital Forensics. His academic interests centre on digital forensics, incident response, and the application of machine learning techniques to security problems. His final year research focused on static malware detection using image-based convolutional neural networks, where he developed practical skills in Python programming, data engineering, and large-scale model evaluation.

Outside his academic work, he has a long-standing involvement in gaming and competitive esports. He is a former top-100 Overwatch player and competed in the Contenders series as a semi-professional, gaining experience in strategy, teamwork, and operating under pressure. He also enjoys a range of outdoor activities including rock climbing, hiking, and golf, and maintains a strong interest in emerging technologies and personal technical projects. He is now developing his professional portfolio as he prepares for a career in cyber security, with particular interest in threat analysis, digital investigations, and the role of artificial intelligence in defensive security.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2026