Robustness of CNN-based malware byteplot classification under standard image transformations

Tudor Ioan Tănăsescu

Supervisors: Tom Viering, Akash Amalan

EEMCS, Delft University of Technology, The Netherlands

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

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

Abstract

CNN-based malware byteplot classifiers achieve high accuracy under clean conditions, but their robustness to input perturbations remains poorly understood. This paper systematically evaluates how a ResNet18 classifier degrades under standard image transformations (rotations, brightness and contrast shifts, and flips) applied to both grayscale and RGB byteplot representations of a 14-class, 20,020 image dataset. Each transformation is grounded in a realistic attacker model via its correspondence to a binary level obfuscation technique. Results show that rotations are catastrophic even at small angles, flips exhibit strong asymmetry driven by the vertical structure of byteplot sections, and photometric shifts are tolerated until a certain threshold. Training on a mixed set that includes 25% distorted images substantially recovers robustness across all categories, with gains being largely self-attributing to their matching transformation type. RGB representations amplify whatever performance trend is already present in grayscale, for better or worse. Together, the findings reveal that CNN byteplot classifiers exploit specific spatial and photometric properties of their input, and that targeted augmented training can harden them against realistic evasion strategies without sacrificing clean image performance.

1 Introduction

Malware, malicious software engineered to disrupt, damage, or gain unauthorised access to digital systems, represents a persistent and escalating threat in the contemporary cybersecurity landscape. The scale of the problem is substantial: the cost of cybercrime in 2020 is estimated at around 1 trillion USD, roughly 50% more than in 2018 [1], and the subsequent decade has seen the emergence of automated malware generation toolchains, polymorphic engines, and AI-assisted obfuscation frameworks that produce thousands of functionally equivalent but structurally distinct variants every day [2]. The consequences extend well beyond individual machines: malware campaigns have disrupted national healthcare infrastructure, compromised financial institutions, and targeted industrial control systems, imposing costs that are simultaneously economic and humanitarian [3].

Traditional signature-based detection, which maintains databases of known malware fingerprints for direct comparison against incoming files, has been progressively outpaced by this volume. Static analysis, which consists of inspecting binary structure without executing code, offers broader coverage but is routinely defeated by obfuscation: packing, encryption, code transposition, and polymorphic rewriting alter the syntactic surface of a binary while preserving its malicious behaviour, breaking signature matches [4].

Dynamic analysis, which executes a suspicious file in an isolated environment and observes its runtime behaviour, is more resilient to structural obfuscation but carries significant drawbacks: it is resource intensive, time consuming, inherently incomplete (malware that detects the sandbox may suppress its payload), and difficult to scale to the volume of new samples encountered in production [5].

CNN-based detection. Against this backdrop, Nataraj et al. introduced a conceptually simple but empirically powerful alternative: representing malware binaries as byteplot images [5]. The raw bytes of an executable are read as unsigned 8-bit integers and arranged into a two dimensional matrix whose width is determined by file size. The resulting grayscale image captures the global byte level structure of the file in visual form. Because malware samples within the same family tend to share large portions of their codebase (a consequence of reusing code to generate new variants), their byteplots exhibit strong intra-family visual similarity in texture layout. The subsequent adoption of convolutional neural networks (CNNs) elevated performance further, as CNNs can automatically learn discriminative features directly from raw pixel values. Remarkably, architectures such as VGG16 and ResNet have achieved accuracy rates approaching or exceeding 98% on standard malware byteplot datasets [3] [4].

Open problem: robustness. However, a critical question remains largely unexamined: how robust are these classifiers to perturbations of their input? CNNs are known to be sensitive to small changes in input images [6], and the features learned for byteplot classification may depend on absolute spatial positions, global intensity statistics, or other properties that an informed attacker could deliberately disrupt. A malware author who knows a defender uses CNN classifiers can attempt to modify their binary in ways that alter the visual appearance of its byteplot while preserving functionality.

Research question. Understanding whether and how standard image transformations degrade classification performance directly informs the design of more resilient detection systems and clarifies what visual properties these models actually exploit. This is the gap that this paper aims to address, by answering the main research question: **How does the accuracy of a CNN-based malware byteplot classifier degrade under standard image transformations, and what can the pattern of degradation reveal about the visual features the network relies on for classification?**

The following sub-questions will guide the investigation:

- **RQ1**: At what transformation magnitude does classification accuracy begin to degrade significantly?
- **RQ2**: Does training on distorted images improve robustness?
- **RQ3**: When trained on a single transformation type, does robustness improvement remain specific to that transformation or generalise to others?
- **RQ4**: Do grayscale and RGB byteplot representations exhibit different robustness profiles?

Below are a few figures to help illustrate the transformations the project will look at. All figures were generated from the same original binary.

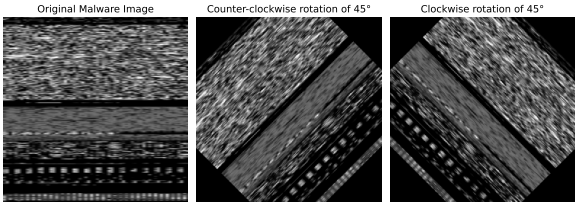

Figure 1: Rotation Example

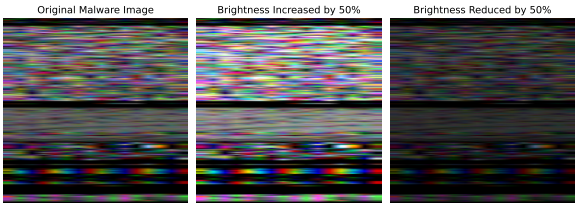

Figure 2: Brightness Example

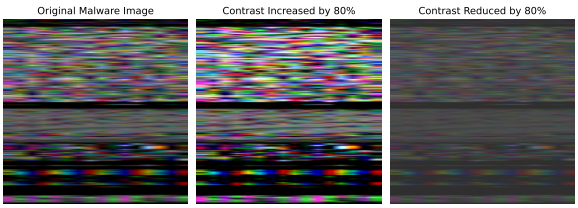

Figure 3: Contrast Example

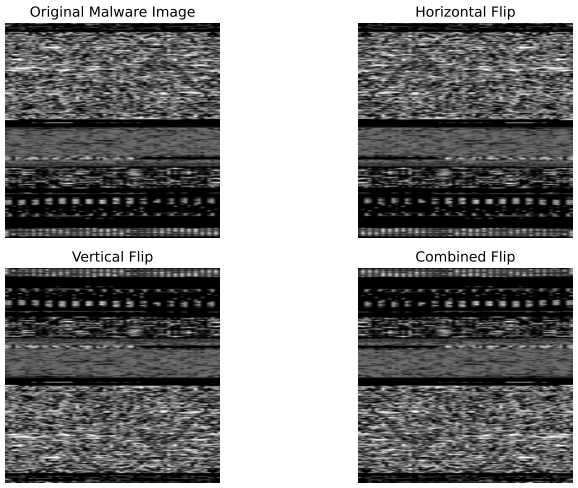

Figure 4: Flips Example

2 Background

This section will discuss the background of the work, divided into two sub-sections. The first will explore related work while the second will establish an attacker model. This is necessary for interpreting the experiments as more than image perturbation studies, giving more purpose to the research beyond the simple robustness of the CNN model tested.

2.1 Related Work

The byteplot approach was introduced by Nataraj et al. [5], who observed that malware families produce visually distinctive grayscale images when their raw bytes are rasterised. Marastoni et al. [4] specifically explored the effect of standard image augmentation techniques during training, including flipping, rotation, and interpolation, on malware classification accuracy, finding that interpolation-based resizing preserves relevant information better than cropping or padding. Their work treats augmentation as a tool for improving classification performance on small datasets, rather than examining how a model trained on clean images responds when presented with transformed inputs at test time.

Brosolo et al. [7] evaluate a CNN classifier against packing with UPX and basic metamorphic transformations, demonstrating accuracy reductions of up to 50%, and use explainability tools, such as HiResCAM, SHAP, and occlusion maps, to diagnose where the classifier's attention shifts after obfuscation. Their analysis reveals that CNN decisions in malware classification often rest on surprisingly localised regions of the byteplot, making the model fragile when those regions are structurally disrupted.

Location of discriminative features. Di Troia et al. [8] conduct a spatial analysis of malware byteplots, training CNN sub-models on individual quadrants of the image and discovering that discriminative features concentrate disproportionately in the bottom left quadrant, likely because malware payload information tends to reside near the end of binary files when rasterised. This conclusion should be read with some caution, since training a sub-model exclusively on one quadrant only allows it to learn from the information available in that quadrant. Strong bottom left performance is consistent with discriminative features concentrating there, but it does not rule out that useful features exist elsewhere and were simply never made available to the other sub-models. They further introduce an image salting technique, replacing a small percentage of pixels with values drawn from another image, and find that even minimal image perturbations, such as pixel modifications affecting under 1% of the total image, drastically degrade accuracy, revealing the fragility of CNNs. However, both this study and the literature cited above do not examine the effect of standard, parameterised image processing operations applied directly to the image independently of the underlying binary. Nor do they investigate how the accuracy degradation curve evolves as transformation magnitude increases, or whether training on specific transformations generalises to robustness against them at inference time.

A computer vision parallel. In the broader computer vision literature, Hendrycks and Dietterich [9] established the ImageNet-C benchmark to systematically evaluate neural network robustness to 15 common image corruptions drawn from four categories (noise, blur, weather, and digital) each applied at five severity levels. Their framework demonstrated that even state of the art classifiers trained on clean ImageNet data experience substantial accuracy degradation under mod-

erate corruption severity, and that data augmentation with corruption matched transforms can recover a portion of this robustness. This paradigm of severity parameterised evaluation across transformation types is the closest methodological analogue to the approach taken in this project, but it has not been applied within the malware byteplot domain.

2.2 Threat Model

The malicious actor. The attacker is modelled as a malware author who is aware that their target environment uses CNN byteplot classification for detection. The attacker's objective is to modify their binary so that it evades the classifier while preserving its malicious payload. This model is consistent with the standard attacker formulation in adversarial malware literature, where Demetrio et al. [10] define the set of admissible manipulations as exactly those that satisfy both syntactic validity (the file must be Portable Executable (PE) or Executable and Linkable Format (ELF), the executable file types for Windows and Linux respectively) and semantic validity (functionality must be preserved).

Attacker knowledge and toolset. The attacker has white box knowledge of the detection paradigm: they know the system operates on byteplots, and understand that visual appearance affects classifier output. They do not have access to the model's weights, architecture, or training data, but they do have the ability to apply binary level transformations.

Transformations as imperfect proxies. A byteplot is a lossless, one-to-one mapping from the byte stream of an executable to a 2D image: each pixel's intensity is exactly the unsigned 8 bit value of the corresponding byte. However, applying standard image transformations to this visual representation does not serve as a perfect proxy for real world binary obfuscation. An image transformation alters pixels globally and uniformly, fundamentally ignoring the rigid, structured nature of executable files. This is a limitation of the study, and therefore the image transformations studied here should be viewed as approximations of specific evasion techniques, rather than exact replicas.

- **Vertical Flip —> PE section re-ordering**: A vertical flip of a byteplot reverses the row order of the underlying binary, by reversing the ordering of equal width byte blocks in the binary stream. In a PE file, the OS loader maps sections to memory according to their virtual addresses recorded in the section table, not according to their physical order on disk. This means sections can be rearranged in any order. As long as the section table entries are updated accordingly, the loaded executable behaves identically. Subroutine reordering permutes the order in which subroutines appear in the executable without altering the execution trace or program logic. However, a naive visual flip remains an imperfect proxy because it would destructively reverse the PE headers situated at the top of the file.
- **Horizontal Flip —> Block level byte reversal**: A horizontal flip reverses the order of bytes within each row of the byteplot, which corresponds to reversing the bytes within each W-byte block of the binary, where W is the image width. Mantovani et al. [11] classify this as the transposition packing scheme, which rearranges the byte order within the program by switching byte values. For this transformation to produce an executable binary, the encoded payload must be accompanied by a decryption stub that reverses the transposition at runtime before transferring control to the decoded section.
- **Combined Flip —> Whole file byte reversal**: This corresponds to reversing the entire byte sequence of the binary. It is a more aggressive form of the transposition encoding described above, requiring a corresponding decryption stub and careful header preservation to maintain semantic validity.
- **Brightness Shifts —> Monotone byte value scaling**: In PyTorch, a brightness adjustment by factor f applies the transformation new_byte = clamp(old_byte × f, 0, 255) to every byte uniformly. This is a monotone, magnitude preserving transformation that shifts the global mean byte value of the binary upward ($f > 1$) or downward ($f < 1$). At the binary level, this corresponds to custom arithmetic encoding, a subclass of monoalphabetic substitution documented by Mantovani et al. [11], in which payload bytes are uniformly scaled by a constant factor before embedding in the packed file, with a corresponding decryption stub that inverts the scaling at runtime. In its simplest form, a brightness decrease by factor 0.5 is mathematically identical to a right bit shift of each byte by one position, which halves every byte value and compresses the byteplot's intensity distribution toward zero. The corresponding decryption stub performs the inverse.
- **Contrast Shifts —> Selective encryption and low entropy encoding**: A contrast increase stretches the range of pixel values toward 255, simulating the byte value spread produced by strong encryption applied to data sections. Conversely, a contrast decrease compresses the byte distribution toward the centre of the intensity range, mirroring the effect of low entropy encoding schemes.
- **Rotations**: Rotating a byteplot at arbitrary angles has no standard binary level counterpart: there is no natural binary operation that corresponds to rotating a two-dimensional representation of a file. They are included as a diagnostic transformation rather than a direct adversarial threat: a classifier that is sensitive to rotation relies on the absolute orientation of spatial features, while a rotation invariant classifier has learned features that are more likely based on local texture statistics rather than global layout. This diagnostic directly informs the interpretation of flip results and the overall characterisation of what visual information the classifier uses.

3 Methodology

This section will detail the project's overall structure and provide more information on the dataset that was used. A subsection will be dedicated to the dataset and another to the experimental setup.

3.1 Dataset

Dataset composition. The dataset used in this work was provided by Akash Amalan and consists of a synthetically generated collection of neutered malware binaries and benign executables. It comprises 20,020 images in total, distributed equally between grayscale and RGB byteplot representations. The images span 14 classes: 12 malware families (e.g. Mirai, Spyware) and 2 benign classes, for PE and ELF file types. Due to ethical considerations, it was decided not to provide the byteplots as they could easily be reverted to malware binaries. However, the Python modules and pipelines used in this project are available at this link.

Benefits of a synthetic dataset. The controlled origin of the dataset provides several experimental advantages. Class balance is maintained by construction, eliminating a common confound in malware classification benchmarks (the Malimg dataset used by Nataraj et al. [5], for example, has a highly skewed class distribution with 2,949 samples in the largest class and as few as 80 in the smallest, which can inflate reported accuracy). Furthermore, all binaries were neutered before image conversion: their executable payloads were disabled to prevent accidental execution.

Removing debug sections. An important preprocessing step is applied before byteplot conversion. Compiled binaries often contain debug sections inserted by the compiler, which carry metadata about symbol names, line numbers, and build configurations. Because deep learning models act as highly efficient pattern matchers, a CNN can trivially learn to identify these debug sections rather than the underlying code structure, leading to artificially inflated accuracy that would not generalise to stripped binaries encountered in production. To prevent this, all binaries in the dataset are preprocessed to remove debug sections before byteplot generation. This forces the model to rely on the structural properties of the file rather than on compiler metadata.

The dataset is partitioned into an 80/20 train/test split, yielding approximately 8,000 training images and 2,000 test images. The same split ratios are used for both RGB and grayscale representations.

3.2 Experimental Setup

Given the computational cost of distorting thousands of images and training multiple model variants, the project utilised Kaggle environments equipped with two NVIDIA T4 GPUs.

Preprocessing pipeline. Prior to input into the model, all byteplot images were resized to 224×224 pixels, the input resolution required by the ResNet18 architecture. Pixel values were then normalised using the ImageNet dataset statistics: channel-wise mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] [12]. All preprocessing was applied identically to grayscale and RGB images and to both training and evaluation sets, with the exception that the perturbations were applied after resizing but before normalisation.

Training configuration. The network was initialised with random weights and trained using the Adam optimiser with a learning rate of 1e-4, and Cross-Entropy was used for the loss function. The models were trained for a total of 30 epochs with a batch size of 64 images. The same hyperparameters are used for all model variants and both image types, ensuring that any performance differences can be attributed to training data composition rather than to hyperparameter variations.

Baseline models. The first stage trains two model variants, one on clean images and one on a mixture of clean and distorted images, repeated for both grayscale and RGB byteplots. The model trained exclusively on clean images serves as the reference against which all robustness comparisons are made. The second model is trained on a mixture of clean and perturbed images. Distorted images constitute one quarter of the training set (equivalent to 20% of the full dataset). The distorted images are distributed uniformly across malware families by stratified sampling: each family receives 143 distorted training images, with approximately 15 images per transformation type per direction (e.g., 15 for brightness increase and 15 for brightness reduction), accounting for the directional nature of non-categorical transformations. For rotations, brightness shifts, and contrast shifts, the step size within the training set is 3 units, with a maximum rotation magnitude of 45 degrees and maximum brightness and contrast shift of 45%.

Single transformation models. The second stage of the project involves training on specific transformation types to pinpoint exactly which transformation in the training set is responsible for any performance gains during evaluation of the model trained on distorted images. Each model in this stage is trained on the same total number of augmented images as in the first (143 per family), but this leads to more images per transformation direction than before (approximately 71-72 versus 15). To keep in line with the general model's training boundary, the step size for non-categorical transformations is adjusted to 0.625 to maintain the same maximum training magnitude of approximately 45 units. This ensures that any performance differences between Stage 1 and Stage 2 models can be attributed to the composition of the training set rather than to its maximum magnitude.

Evaluation. All models are then evaluated against the test set subjected to each transformation type. Rotations are evaluated from 0° to 180° in 2° increments, applied in both clockwise and counter-clockwise directions to detect any directional asymmetry. Brightness and contrast shifts are evaluated to a magnitude of 60% in 2% increments in both directions. Very good performance was observed for the model trained on perturbed images in Stage 1, especially for brightness and contrast shifts, so for Stage 2 the maximum magnitude during evaluation was set to 100% for these transformations. Flips are evaluated as three categorical conditions: horizontal only, vertical only, and combined (horizontal and vertical simultaneously). Classification accuracy is recorded at every evaluation point, producing high resolution degradation curves.

4 Results

This section will first present the classification performance of the general models evaluated under standard image transformations, then it will look at the models trained on specific transformations. A comparison of the results is reserved for the Discussion section.

4.1 General Models

Both models achieved almost 100% accuracy for the baseline evaluation on non-distorted images, for both grayscale and RGB byteplots, the worst performing model having a 99.95% accuracy. Any result below 90% will be considered inadequate. This threshold marks a clear, double digit drop from the almost perfect clean baseline.

These baseline results establish that the ResNet18 architecture is highly capable of classifying the clean byteplots, and that substituting 20% of the training set with distorted images does not incur a penalty on clean-image evaluation.

The model trained on **non-distorted** images is the overall baseline for the project. The results that follow are the template that will be used to judge whether training on distorted images provides any benefit.

Rotations are by far the most destructive transformations, with even a small angle of 10 degrees halving accuracy (Table 1). The degradation patterns were found to be largely symmetrical regardless of the rotation direction, but RGB images showed worse performance earlier than grayscale images.

Table 1: Accuracy on rotated images, model trained on clean images

Rotation Type	Image Type	4°	10°	90°
Counter-Clockwise	Grayscale	0.86	0.48	0.22
Clockwise	Grayscale	0.88	0.49	0.24
Counter-Clockwise	RGB	0.86	0.31	0.19
Clockwise	RGB	0.85	0.38	0.21

Brightness and **contrast** shifts were tolerated until a certain threshold, after which accuracy dropped sharply. RGB images perform similarly, although the threshold at which accuracy begins to drop occurs earlier and it is more consistent for all transformations, whereas for grayscale each transformation has its own, unique threshold (Figure 5).

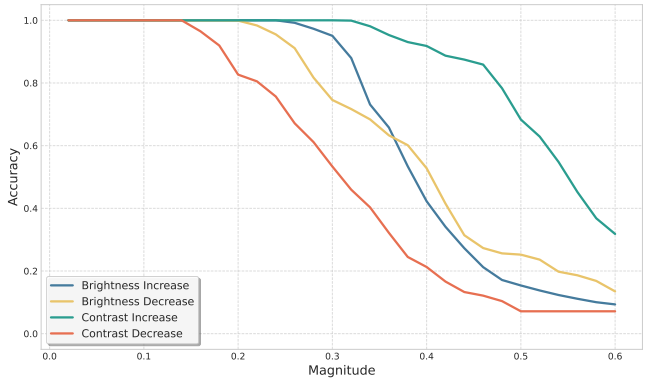

Figure 5: Brightness and contrast shifts, model trained on clean grayscale images

Horizontal **flips** presented surprisingly good results, while vertical flips proved hard for the model to classify, and the same is true for combined flips. RGB images accentuate the results from grayscale here: where there is high performance there are even better results, where there is lower performance there are even worse results (Table 2).

Table 2: Flips, model trained on clean images

Flip Type	Image Type	Accuracy
Horizontal	Grayscale	0.96
Vertical	Grayscale	0.40
Combined	Grayscale	0.37
Horizontal	RGB	0.99
Vertical	RGB	0.29
Combined	RGB	0.30

The model trained on **distorted** images showed improved performance for **brightness** and **contrast shifts**. For RGB images, all transformations maintain around 95% accuracy or above for the entire evaluation phase, while for grayscale only contrast decreases dip below this value, reaching an accuracy slightly below 60% for this particular transformation (Figure 6).

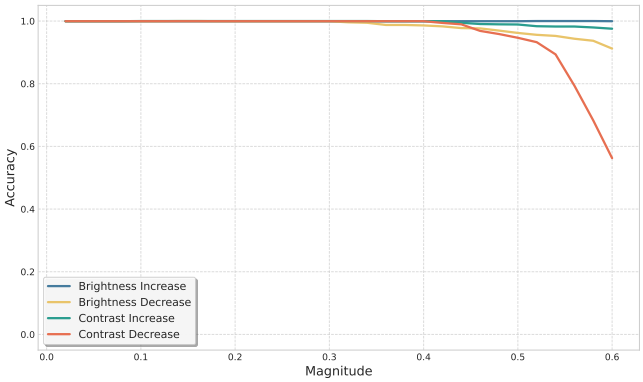

Figure 6: Brightness and contrast shifts, model trained on distorted grayscale images

Rotations remain the most destructive transformations. The highest improvements are seen only until the training boundary of 45°, after which a smaller but still significant uplift is observed (Table 3).

Table 3: Accuracy on rotated images, model trained on distorted images

Rotation Type	Image Type	4°	10°	90°
Counter-Clockwise	Grayscale	0.83	0.69	0.41
Clockwise	Grayscale	0.87	0.73	0.40
Counter-Clockwise	RGB	0.85	0.75	0.28
Clockwise	RGB	0.86	0.75	0.41

All **flip** types for both grayscale and RGB images showed almost 100% accuracy. In both cases, the large disparity between flip types observed in the clean model is eliminated.

4.2 Models Trained on One Transformation

The model trained only on **brightness** shifts unsurprisingly accounts for essentially all of the brightness robustness observed in the general augmented model, and partially explains its improved tolerance to contrast increases, but contributes nothing to the performance gains seen on flips, contrast decreases, or rotation. This is true for both image types, with RGB showing even better performance, with 100% accuracy for contrast increases throughout the evaluation (Figure 7).

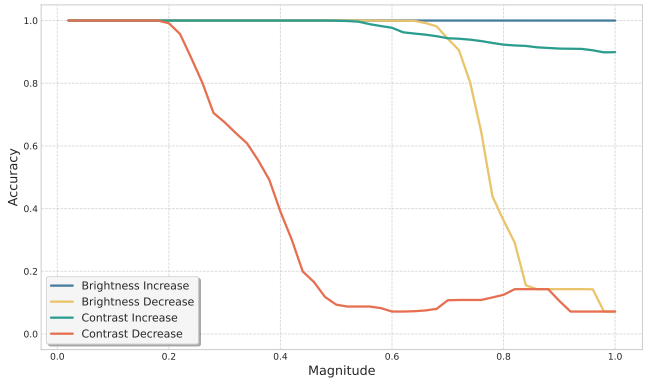

Figure 7: Brightness and contrast shifts, model trained only on brightness shifts, grayscale images

Contrast only training fully accounts for the general model's robustness to contrast increases, with both image types maintaining 100% accuracy throughout. It also accounts for the gains related to contrast reduced images. No contribution to flip or rotation improvements is observed, but a modest transfer to brightness increases is visible within the evaluated range (Figure 8).

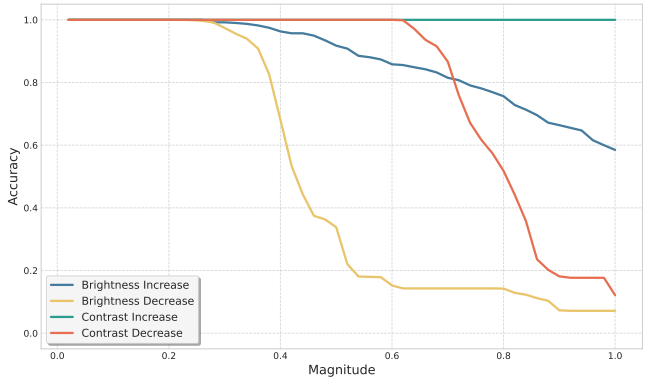

Figure 8: Brightness and contrast shifts, model trained only on contrast shifts, RGB images

Rotation only training accounts for most of the general model's improved robustness to this transformation, with both image types sustaining above 70% accuracy through 90° where the general model collapsed. It contributes nothing to brightness or contrast robustness, and does not reproduce the general model's near-complete recovery at 180°. A more interesting result lies in the accuracy for flips (Table 4), where we can see a clear uplift in performance for vertical and combined flips, even though the model was trained strictly on rotations of up to 45 degrees.

Table 4: Flips, model trained only on rotations

Flip Type	Image Type	Accuracy
Horizontal	Grayscale	0.97
Vertical	Grayscale	0.64
Combined	Grayscale	0.63
Horizontal	RGB	0.93
Vertical	RGB	0.54
Combined	RGB	0.49

Training exclusively on **vertical flips** fully resolves sensitivity for this transformation in both image types. The more notable finding is that it also accounts for nearly all of the general model's combined flip recovery and, by geometric equivalence, its 180° rotation recovery, with accuracy reaching 98% (grayscale) and 94% (RGB) on both transformations. Mid-range rotation, brightness, and contrast robustness are unaffected.

Horizontal flip training resolves sensitivity for this transformation completely, reaching 100% in both image types, but unlike vertical flip training, it transfers to nothing else.

Finally, **combined** flip training resolves vertical and horizontal flip sensitivity completely in both image types and delivers near-perfect 180° rotation recovery, with 99.9% accuracy in grayscale and 100% in RGB. As expected, mid-range rotation, brightness, and contrast robustness are unaffected.

5 Discussion

This section will interpret the results presented above, taking into account the four research questions outlined in the introduction.

5.1 RQ1: Degradation Thresholds and Their Variation Across Transformation Types

Rotation and spatial anchoring. Rotations are catastrophic even at small magnitudes. A rotation of 10° more than halves accuracy for the clean grayscale model and reduces it to below 40% for RGB. This extreme sensitivity indicates that the classifier has learned features that are tightly coupled to absolute spatial orientation: the global layout of the byteplot encodes structural information that the network exploits. The moment the image is rotated, these globally positioned features no longer occupy their expected spatial locations, and the classifier's accuracy collapses.

Flip asymmetry and row structure. While horizontal flips are surprisingly well tolerated, vertical and combined flips show significant degradation in performance. A horizontal flip preserves the semantic structure of the byteplot, while a vertical flip reverses the row ordering entirely, placing what was the file's tail at the top and the header at the bottom, which constitutes a complete disruption of the structural information the classifier relies on. The fact that horizontal flip

performance is high even without training on distorted images suggests the model has implicitly learned features that are symmetric or approximately symmetric within rows.

Brightness and contrast shifts exhibit a qualitatively different pattern: a plateau of almost perfect accuracy followed by a sharp threshold beyond which performance drops rapidly. This suggests the classifier is tolerant of photometric perturbations up to the point where the global intensity distribution of the image shifts enough. Essentially, the network's learned filters still respond correctly to scaled or shifted byte values within a range, but once the magnitude of the shift pushes the distribution far enough outside what was covered by training data, performance begins to drop quickly.

5.2 RQ2: Does Training on Distorted Images Improve Robustness?

Augmentation improves robustness broadly. Training on a mixed set that includes distorted images consistently and substantially improves robustness across all transformation categories, confirming that the accuracy degradation observed in the clean model is not an intrinsic property of byteplot classification. The improvement is most striking for flips, where the augmented model achieves near perfect accuracy across all three flip types (97–99%) in both image representations, compared to as low as 29% for vertical flips in the clean RGB model. This recovery indicates that the visual features sufficient to distinguish malware families under flipped conditions exist in the byteplot and can be learned when the training distribution includes them, but the clean model simply never encounters these orientations.

Strong photometric recovery, with one exception. For brightness and contrast shifts, the augmented model demonstrates strong robustness especially for RGB images, which maintain close to or above 95% accuracy throughout the evaluation range for all photometric transformations. The grayscale model under contrast decreases represents the one notable exception, dipping below 60%. This suggests that for grayscale images, contrast reduction compresses the intensity sufficiently to eliminate features that cannot be compensated for with only 25% of the training set being distorted.

Rotations are still the hardest case. Rotations remain the most challenging transformation even with augmentation. While accuracy at 10° improves from approximately 40–50% to 69–75%, performance still degrades substantially through the evaluation range, and the improvement is considerably worse after the 45° training boundary: gains beyond this threshold are smaller in magnitude than those observed within it. This implies that the spatial disorientation caused by arbitrary rotations is only partially addressed by training with a limited set of angles. The classifier learns a degree of tolerance but does not achieve full rotation invariance, which would likely require training with a much denser coverage of the rotation space.

5.3 RQ3: Attribution of Robustness Gains to Specific Transformations

Robustness gains are largely self-attributing. The single transformation training experiments reveal a clear pattern: robustness improvements are primarily self-attributing, meaning training on a given transformation accounts for most of the gains to that same transformation in the general model. However, in two instances they work together.

Rotation to flip transfer. The first is also the simplest to explain. Due to the geometrical equivalence of a combined horizontal and vertical flip and a 180° rotation, training on vertically flipped or combined flipped images improves performance for rotations at or close to 180°. Training exclusively on rotations produces partial robustness to vertical and combined flips, and the accuracy values on the combined flip and the 180° rotation evaluation are numerically identical (as they should be due to their geometric equivalence). The transfer from rotation training to vertical flip robustness is explained by the fact that exposure to a range of rotations up to 45° prevents the model from anchoring features purely to their absolute vertical position, and this partial orientation tolerance extends to the vertical component of a flip.

An asymmetric transfer between flips. Another significant finding is that vertical flip training transfers strongly to combined flip and 180° rotation robustness, but horizontal flip training transfers to nothing else. This asymmetry mirrors the asymmetry in the original degradation results and strongly suggests that vertical structure is the primary carrier of discriminative information in byteplots, and that learning to cope with its disruption is a powerful generalisation.

A modest brightness-contrast correlation. The second instance of a positive correlation is for the photometric transformations. Although it is more modest, a real correlation between brightness and contrast is observed: brightness training partially improves contrast increase tolerance, and contrast training provides modest gains on brightness increases. This is consistent with the observation that both transformation types modify the global intensity distribution of the image, so a model trained to handle one form of photometric shift acquires a partial generalisation to related shifts. However, this transfer is somewhat one directional: contrast training provides better brightness gains than brightness training provides contrast gains, which may reflect that contrast changes are a more powerful perturbation.

5.4 RQ4: Grayscale vs RGB Robustness

RGB amplifies existing trends. The comparison between grayscale and RGB images reveals a consistent amplification pattern: the RGB representation accentuates whatever trend is already present in the grayscale results. Where grayscale performance is high, RGB is higher. Where grayscale performance is low, RGB is lower. For horizontal flips, where both image types perform well even without augmentation, the RGB model achieves 0.99 accuracy compared to 0.96 for grayscale. For vertical flips under the clean model, however, RGB accuracy (0.29) falls well below grayscale (0.40). A similar pattern holds for rotations: RGB degrades faster than grayscale at small rotation angles, even though the augmented RGB model eventually reaches comparable or slightly better performance.

The practical implication is that the choice between grayscale and RGB representations involves a trade-off: RGB images provide stronger clean performance and stronger augmented performance in most categories, but are more frag-

ile when transformations fall outside the training boundary. RGB based classifiers therefore have more to gain from perturbed images in the training set.

5.5 Implications for Malware Detection

Structure over content. Read together, the results for rotations and flips point to the same underlying mechanism: the classifier relies heavily on where certain byte patterns conventionally appear. The fact that the network collapses under rotation and vertical flips, yet tolerates horizontal flips, is best explained by a model that has learned vertical position as a proxy for file structure rather than learning the content itself. As noted in the introduction, malware families re-use a lot of code for faster generation of slightly different binaries to avoid detection. The results above suggest it is the layout consequences of that reuse, not the logic, that the network relies on. The fact that the model relies so heavily on the vertical positions of the file's sections leaves it very susceptible to simple section re-ordering, a cheap binary level obfuscation that attackers employ. This means any CNN-based malware classifier should be trained on such binaries.

Byte statistics over byte values. The pattern observed for brightness and contrast shifts is similarly informative. It indicates that the network relies on relative intensity, such as local contrast and texture, rather than on absolute byte values, since moderate uniform scaling is absorbed almost without cost.

Finally, the fact that training with distorted images recovers robustness across every category without harming clean accuracy, and that the recovery is largely traceable to the matching transformation, indicates that this fragility is a property of the training distribution rather than a hard limit on what byteplots can express. The discriminative information needed to survive section reordering or moderate repacking already exists in the image, the clean model simply never sees it during training. This makes training on perturbed images less of an optional refinement and more of a precondition for any sort of actual use.

6 Conclusions and Future Work

Summary of findings. This paper investigated how the accuracy of a ResNet18-based malware byteplot classifier degrades under standard image transformations, and what the pattern of that degradation reveals about the visual features the network relies on. The overarching conclusion is that CNN byteplot classifiers are highly capable under clean conditions but exhibit structured, interpretable vulnerabilities to standard image perturbations. These vulnerabilities directly map onto realistic binary level obfuscation strategies available to a motivated attacker. Crucially, they can be substantially mitigated through targeted training on distorted images without sacrificing clean image performance.

Future work: validation on other datasets. Several directions for future work emerge naturally from the results and the limitations of this study. The most obvious is validation on different datasets. This work uses a synthetically generated, class balanced collection of neutered binaries. While neutering is necessary for safe processing, they may differ structurally from live malware samples in ways that affect the byteplot appearance and the generalisation of these findings. Validation against established benchmarks such as the Malimg dataset would strengthen confidence in the generalisability of the results, even at the cost of dealing with its known class imbalance. Differences in degradation curves across datasets would also clarify whether the specific class distribution, binary origins, or synthetic generation process significantly affect the current results.

Genuine binary obfuscation. A further direction for future work is to apply genuine binary level obfuscation techniques rather than relying on image transformations as proxies. This would shift the evaluation from CNN robustness under proxy perturbations to classification performance on obfuscated code. Such an approach, informed by the findings presented here, could validate whether the vulnerabilities identified in this study generalise to real world obfuscated malware, and may surface additional CNN behaviours that the image level proxies do not capture.

Other architectures. Likewise, studying different architectures would establish whether the fragilities observed are specific to ResNet18 or reflect a broader property of CNN-based malware classification.

Augmentation ratios and targeting. Another possible direction involves the number of distorted images in the training set. For this project, it was constant at 20% of the overall dataset. Increasing this number might lead to even better performance, so it should be taken into account for future studies. The transformations were also applied uniformly for each family. Exploring more targeted strategies would determine whether the robustness gains observed here can be achieved more efficiently or extended to larger transformation magnitudes.

Explainability tools. Finally, explainability tools such as Grad-CAM, HiResCAM, and SHAP could be applied here to directly visualise how the classifier's attention maps change as transformation magnitude increases. Such analysis could directly test the hypothesis that rotation and vertical flip sensitivity are driven by specific regions in the upper portion of the byteplot.

7 Responsible Research

Reproducibility. All preprocessing steps, model hyperparameters, training configurations, and evaluation methods are reported in full in Section 3. The Python modules and pipelines used in this project are made public at this link. The byteplot images themselves are not released: because a byteplot is a lossless, reversible encoding of an executable's raw bytes, distributing the dataset would be functionally equivalent to distributing the underlying binaries.

Data handling and safety. All binaries in the dataset were neutered prior to image conversion, disabling their executable payloads to prevent accidental execution at any stage of the pipeline. Debug sections were stripped before byteplot generation, both to prevent the classifier from learning compiler metadata as a shortcut feature and to reduce the amount of potentially identifying build information retained in the images. The dataset is synthetically generated and class-balanced by construction, and contains no personal or sensitive data about

individuals, so no privacy considerations apply.

Ethical justification. The primary ethical justification for this research lies in its potential to help honest actors protect themselves from malicious parties. Today, society is dependent on secure digital infrastructure, and malware attacks routinely threaten critical sectors such as healthcare networks, financial institutions, and power grids, causing severe economic and human disruption. By investigating the robustness of CNNs and proving that models can be hardened through training on augmented data, this research directly contributes to the development of more resilient defence systems. Conversely, any project that investigates the vulnerabilities of security systems carries an inherent dual use risk. Attackers might find new patterns to exploit, or they can use the results of newly published research in the domain to preemptively modify their malware in order to counteract new defences before they come online. However, these considerations should not stop researchers from investigating new methods of defeating malicious actors and publishing their results, since defenders benefit from this knowledge at least as much as attackers do, and typically have first-mover access to it through the publication process itself.

Use of Generative AI. In compliance with TU Delft regulations, generative AI was used exclusively for correcting grammatical errors and improving text clarity, not for generating new ideas or paragraphs in the text. The prompts had the following patterns:

- ”Check the paragraph below for any grammatical mistakes and point them out”
- ”Read the paragraph below and give me feedback on its readability”
- ”Is this paragraph too casual for an academic paper?”

References

[1] F. Cremer, B. Sheehan, M. Fortmann, A. N. Kia, M. Mullins, F. Murphy, and S. Materne, "Cyber risk and cybersecurity: a systematic review of data availability," *The Geneva Papers on Risk and Insurance - Issues and Practice*, vol. 47, no. 3, p. 698–736, Feb. 2022. [Online]. Available: http://dx.doi.org/10.1057/s41288-022-00266-6

[2] M. Saxena and T. Das, "Hierarchical malware detection, family identification, and variant attribution using CNN-based hybrid models on grayscale executable images," *Scientific Reports*, vol. 16, no. 1, Feb. 2026. [Online]. Available: http://dx.doi.org/10.1038/s41598-026-40655-8

[3] N. Younas, S. Riaz, S. Ali, R. Khan, F. Ali, and D. Kwak, "Detecting malicious code variants using convolutional neural network (CNN) with transfer learning," *PeerJ Computer Science*, vol. 11, p. e2727, Apr. 2025. [Online]. Available: http://dx.doi.org/10.7717/peerj-cs.2727

[4] N. Marastoni, R. Giacobazzi, and M. Dalla Preda, "Data augmentation and transfer learning to classify malware images in a deep learning context," *Journal of Computer Virology and Hacking Techniques*, vol. 17, no. 4, p. 279–297, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1007/s11416-021-00381-3

[5] 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*, ser. VizSec '11. ACM, Jul. 2011, p. 1–7. [Online]. Available: http://dx.doi.org/10.1145/2016904.2016908

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013. [Online]. Available: https://arxiv.org/abs/1312.6199

[7] M. Brosolo, V. Puthuvath, and M. Conti, "The Road Less Traveled: Investigating Robustness and Explainability in CNN Malware Detection," *arXiv preprint arXiv:2503.01391*, 2025. [Online]. Available: https://arxiv.org/abs/2503.01391

[8] A. Roy and F. Di Troia, "Discriminative Regions and Adversarial Sensitivity in CNN-Based Malware Image Classification," *Electronics*, vol. 14, no. 19, p. 3937, Oct. 2025. [Online]. Available: http://dx.doi.org/10.3390/electronics14193937

[9] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," *arXiv preprint arXiv:1903.12261*, 2019. [Online]. Available: https://arxiv.org/abs/1903.12261

[10] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware," *IEEE Transactions on Information Forensics and Security*, vol. 16, p. 3469–3478, 2021. [Online]. Available: http://dx.doi.org/10.1109/TIFS.2021.3082330

[11] A. Mantovani, S. Aonzo, X. Ugarte-Pedrero, A. Merlo, and D. Balzarotti, "Prevalence and Impact of Low-Entropy Packing Schemes in the Malware Ecosystem," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, Jan. 2020. [Online]. Available: https://dx.doi.org/10.14722/ndss.2020.24297

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. [Online]. Available: https://dx.doi.org/10.1109/CVPR.2009.5206848

Appendix

Below is a list of figures showcasing the results of all evaluation phases throughout this project.

Model trained on clean grayscale images

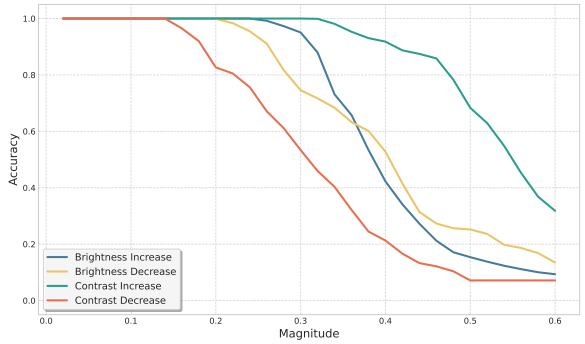

Figure 9: Brightness and contrast shifts

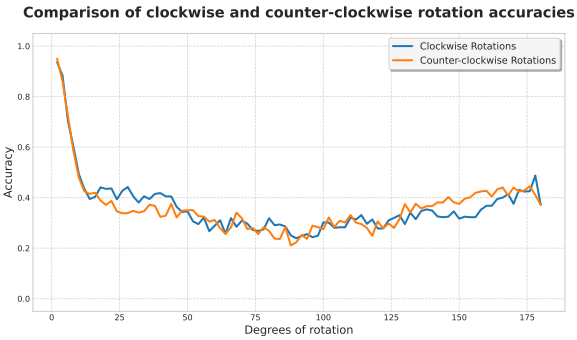

Figure 10: Rotations

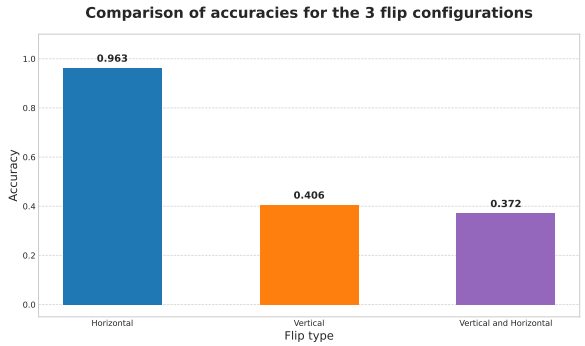

Figure 11: Flips

Model trained on clean RGB images

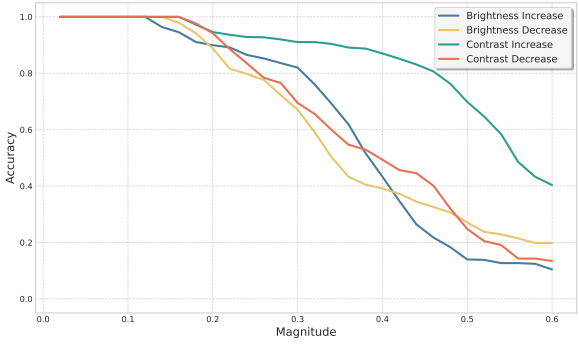

Figure 12: Brightness and contrast shifts

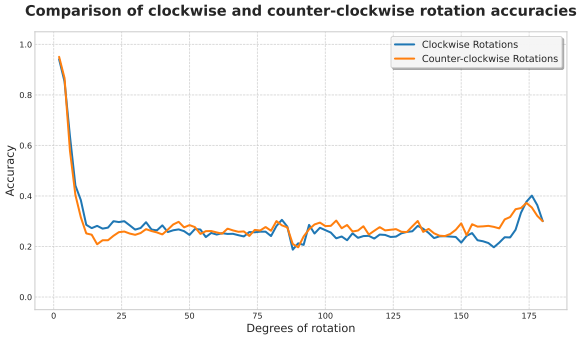

Figure 13: Rotations

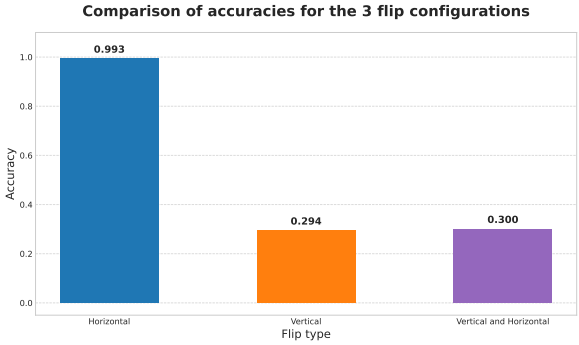

Figure 14: Flips

Model trained on distorted grayscale images

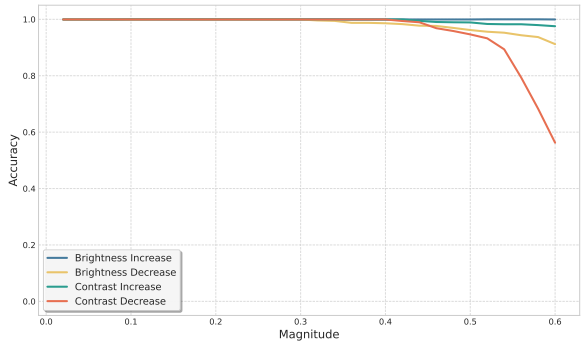

Figure 15: Brightness and contrast shifts

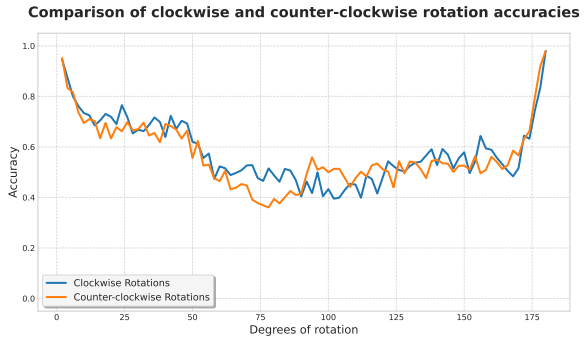

Figure 16: Rotations

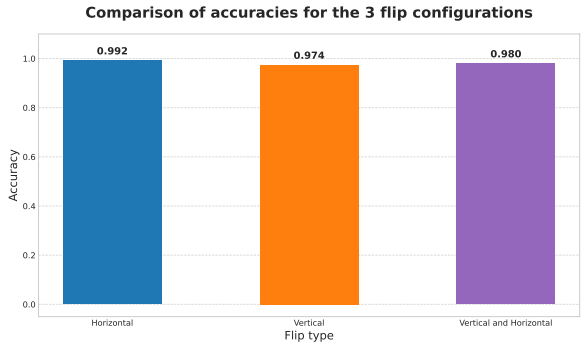

Figure 17: Flips

Model trained on distorted RGB images

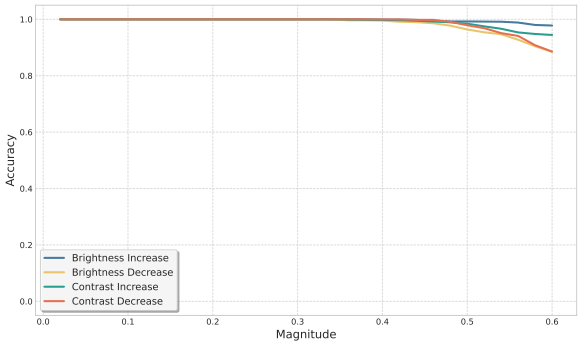

Figure 18: Brightness and contrast shifts

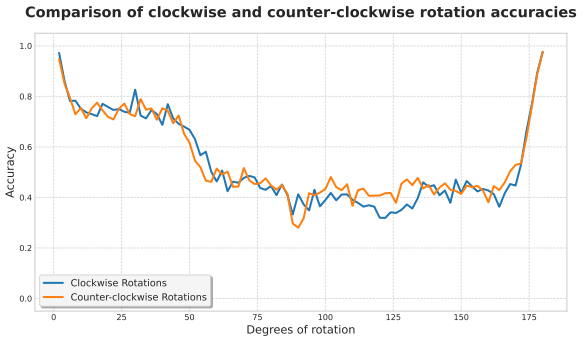

Figure 19: Rotations

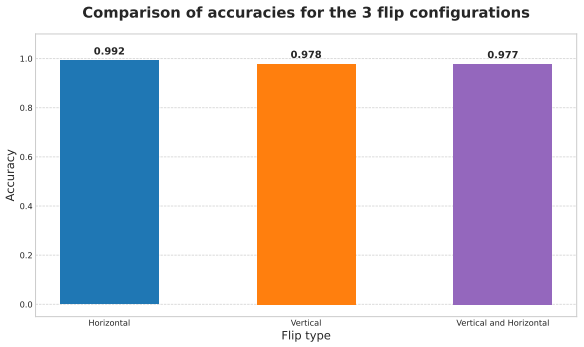

Figure 20: Flips

Model trained only on brightness distorted grayscale images

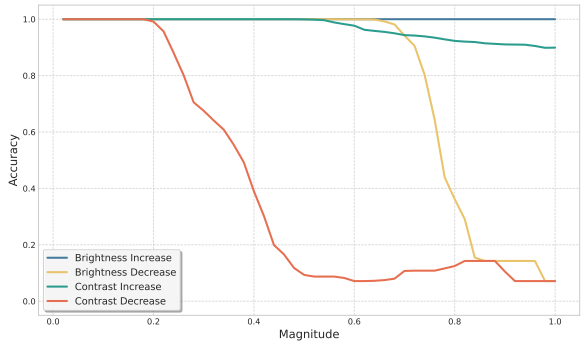

Figure 21: Brightness and contrast shifts

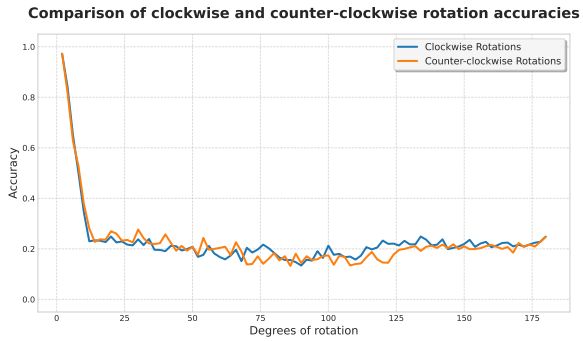

Figure 22: Rotations

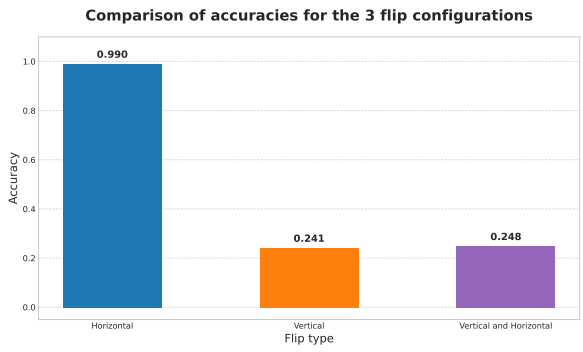

Figure 23: Flips

Model trained only on brightness distorted RGB images

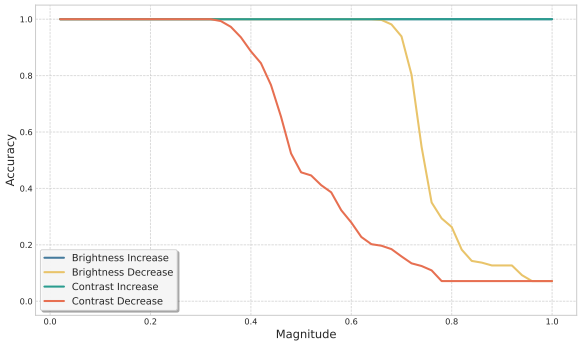

Figure 24: Brightness and contrast shifts

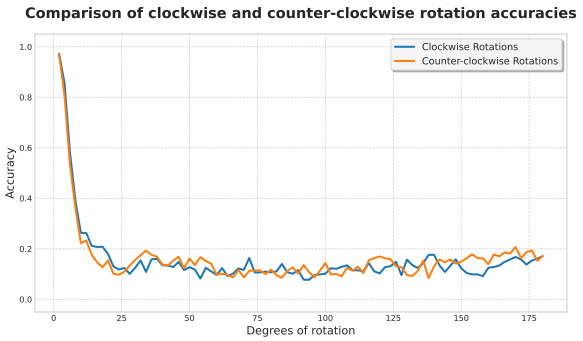

Figure 25: Rotations

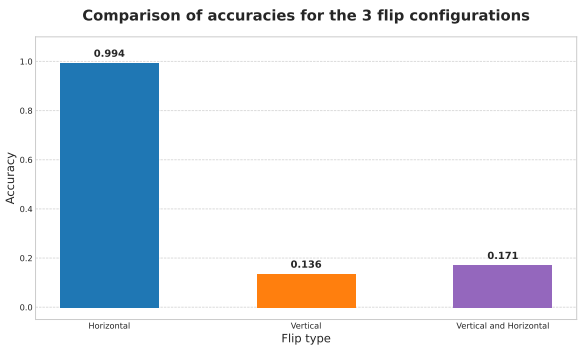

Figure 26: Flips

Model trained only on contrast distorted grayscale images

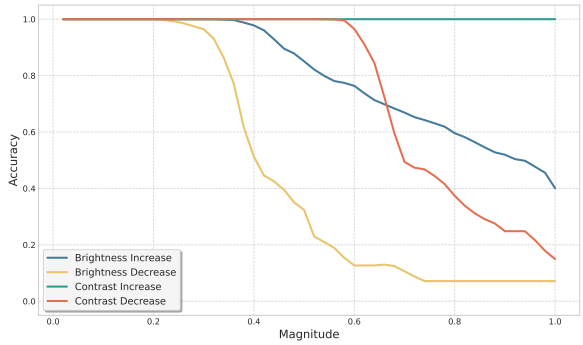

Figure 27: Brightness and contrast shifts

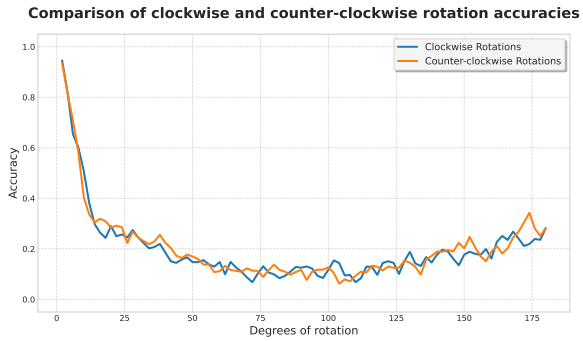

Figure 28: Rotations

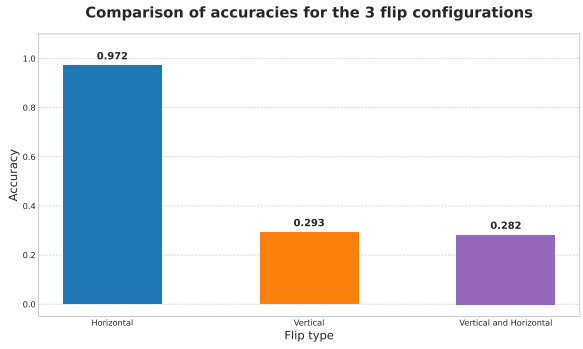

Figure 29: Flips

Model trained only on contrast distorted RGB images

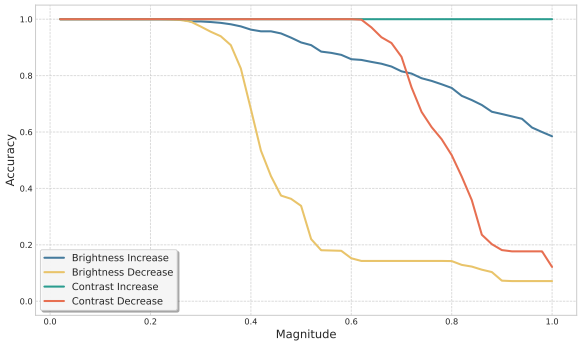

Figure 30: Brightness and contrast shifts

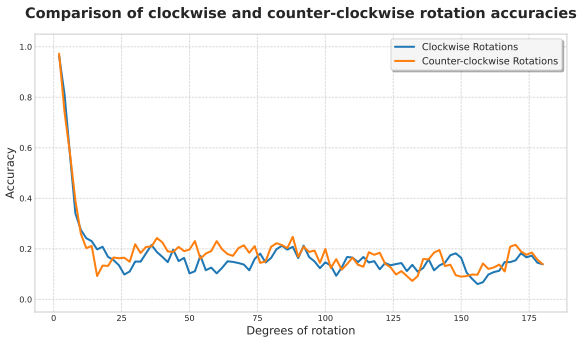

Figure 31: Rotations

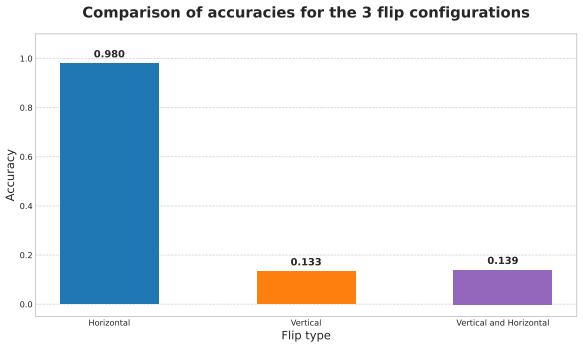

Figure 32: Flips

Model trained only on rotation distorted grayscale images

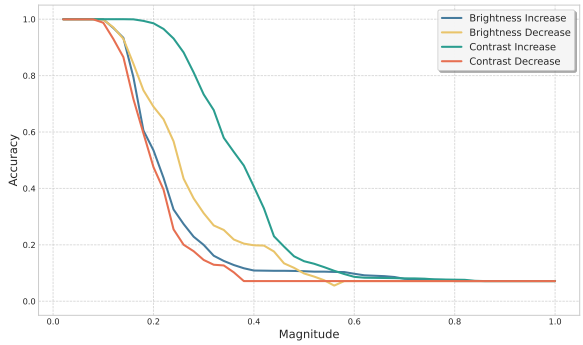

Figure 33: Brightness and contrast shifts

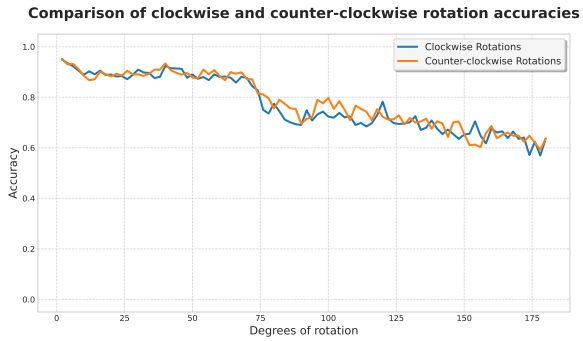

Figure 34: Rotations

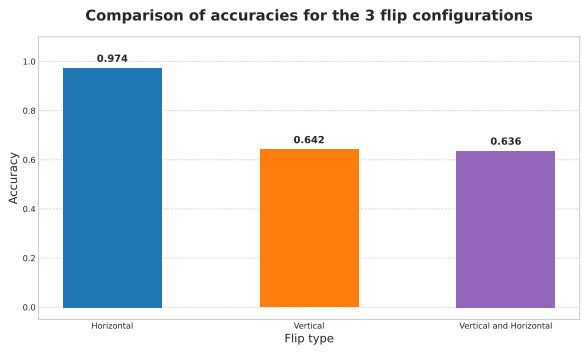

Figure 35: Flips

Model trained only on rotation distorted RGB images

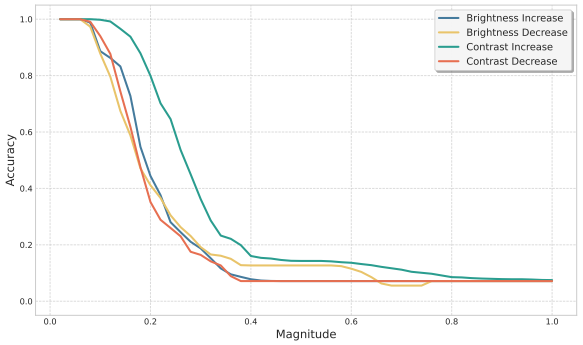

Figure 36: Brightness and contrast shifts

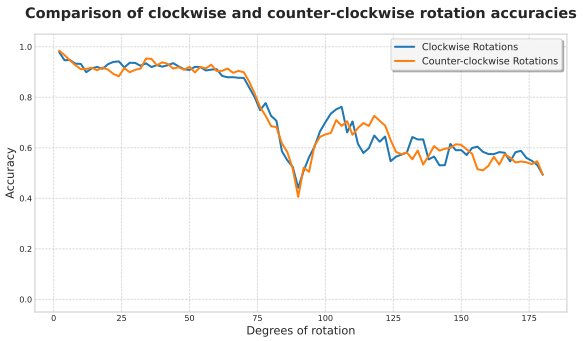

Figure 37: Rotations

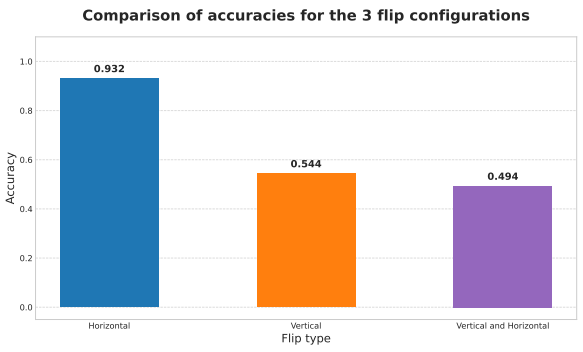

Figure 38: Flips

Model trained only on horizontally flipped grayscale images

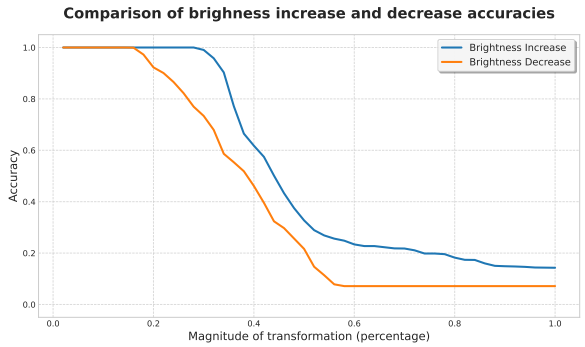

Figure 39: Brightness

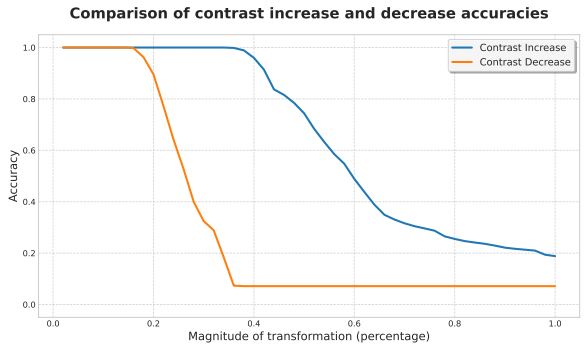

Figure 40: Brightness

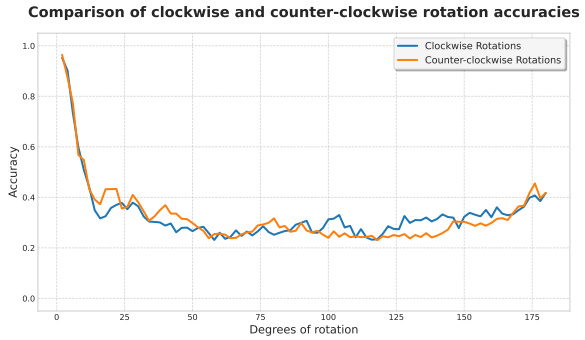

Figure 41: Rotations

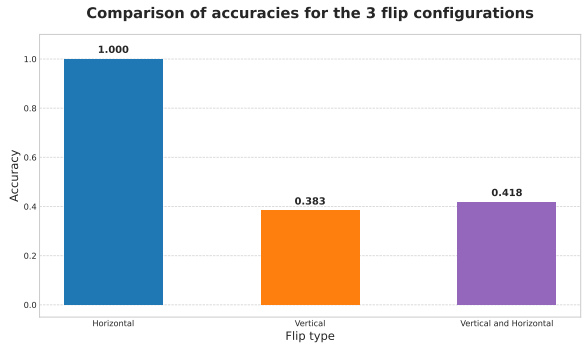

Figure 42: Flips

Model trained only on horizontally flipped RGB images

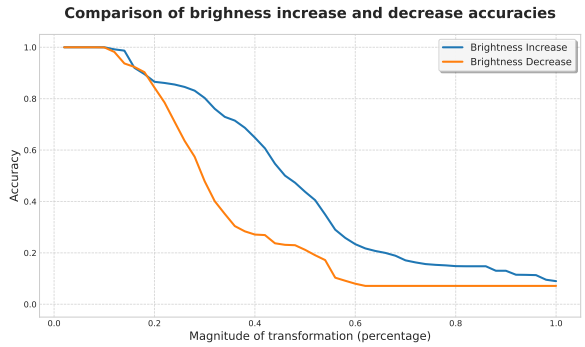

Figure 43: Brightness

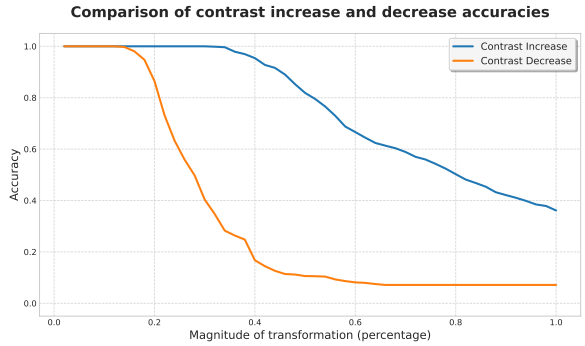

Figure 44: Brightness

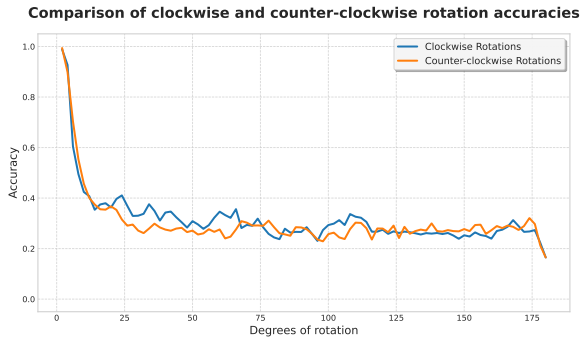

Figure 45: Rotations

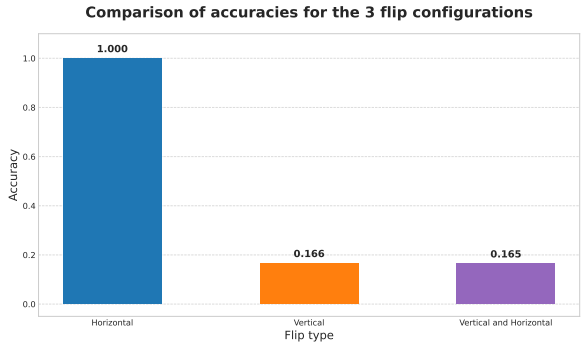

Figure 46: Flips

Model trained only on vertically flipped grayscale images

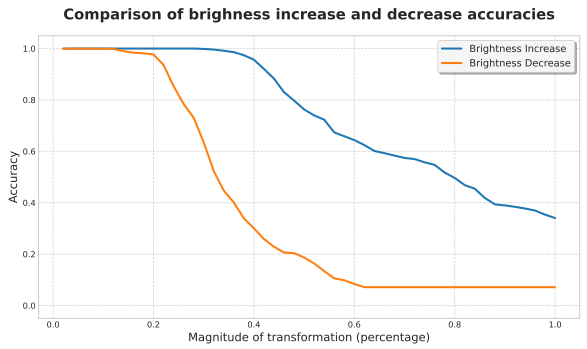

Figure 47: Brightness

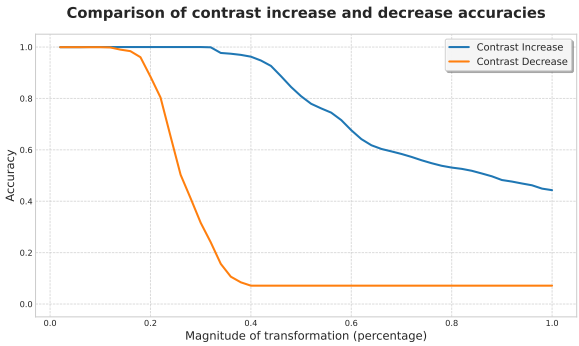

Figure 48: Brightness

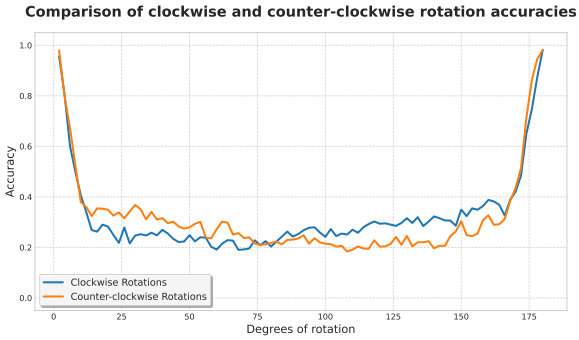

Figure 49: Rotations

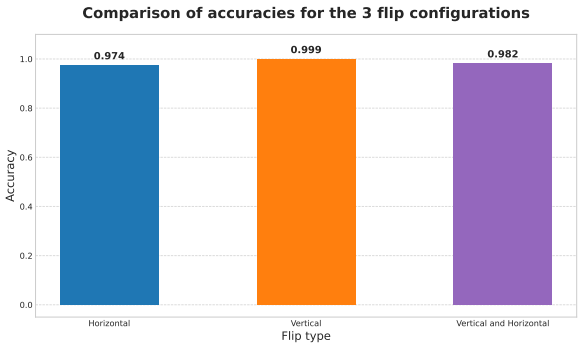

Figure 50: Flips

Model trained only on vertically flipped RGB images

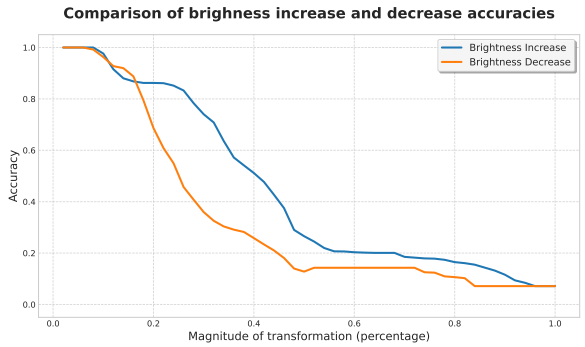

Figure 51: Brightness

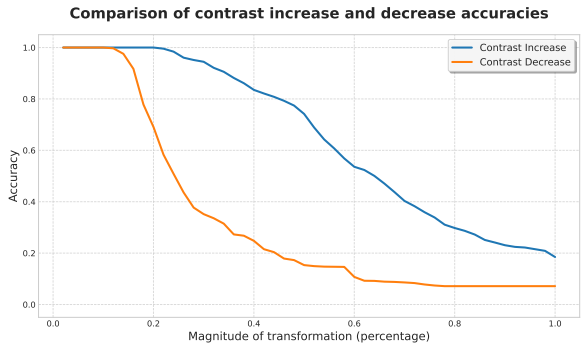

Figure 52: Brightness

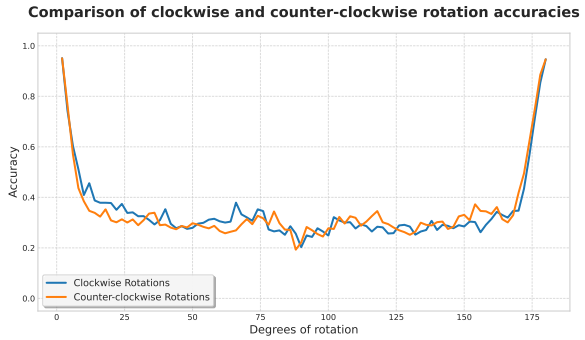

Figure 53: Rotations

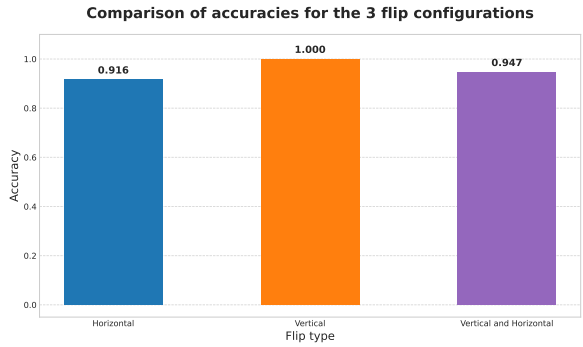

Figure 54: Flips

Model trained only on vertically and horizontally flipped grayscale images

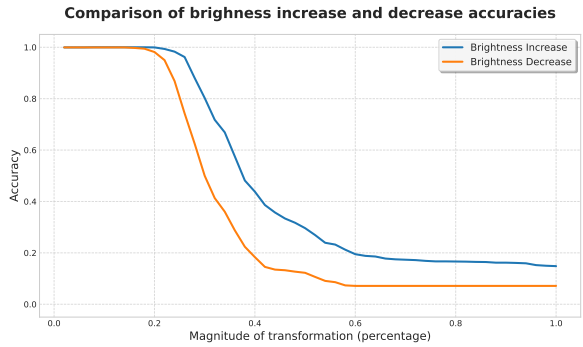

Figure 55: Brightness

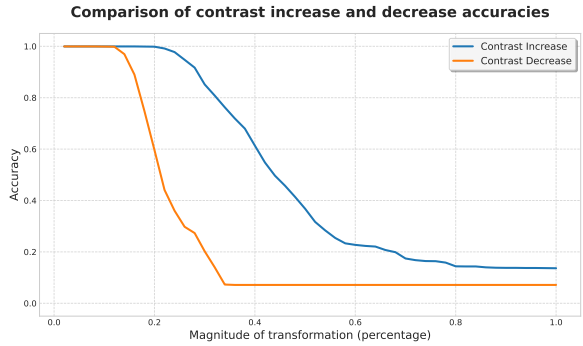

Figure 56: Brightness

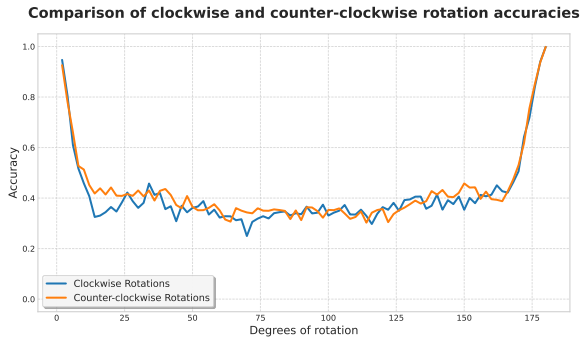

Figure 57: Rotations

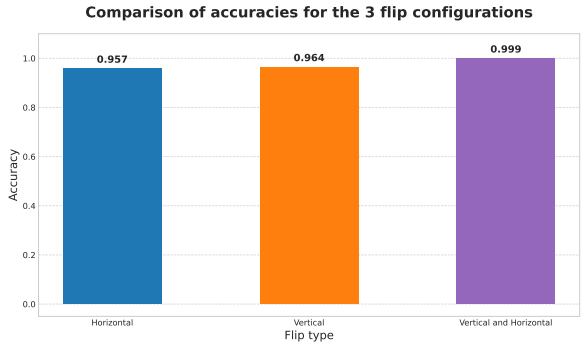

Figure 58: Flips

Model trained only on vertically and horizontally flipped RGB images

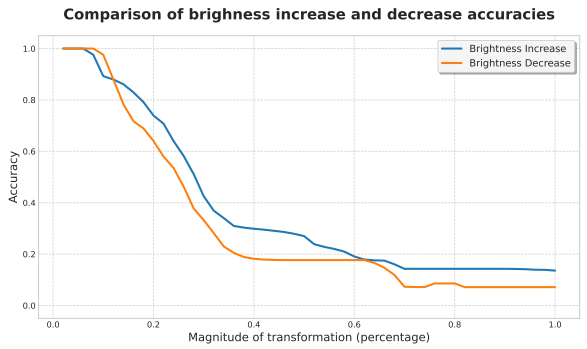

Figure 59: Brightness

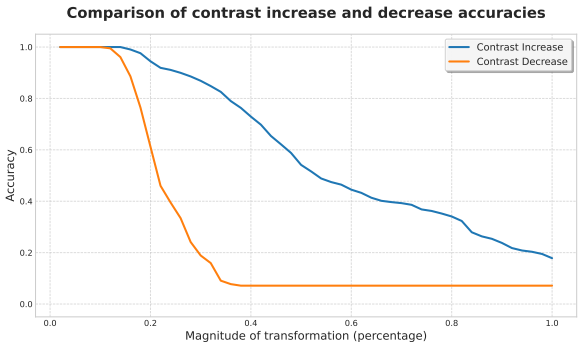

Figure 60: Brightness

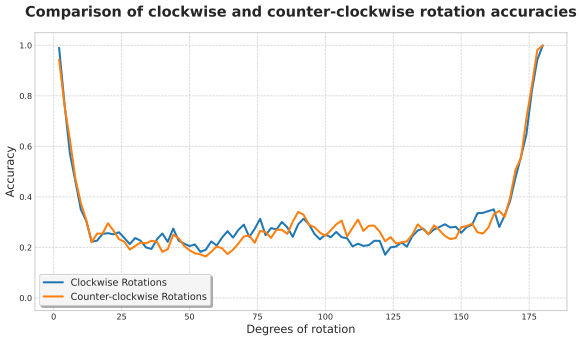

Figure 61: Rotations

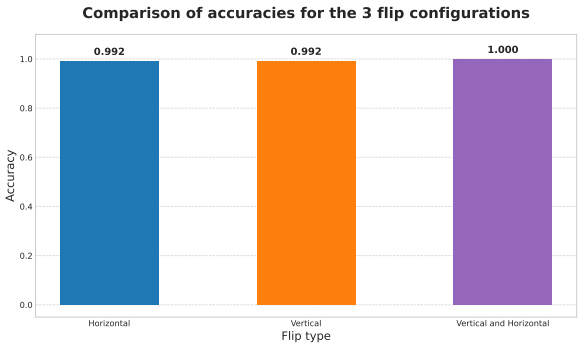

Figure 62: Flips