On the Interplay of Convolutional Padding and Adversarial Robustness

Paul Gavrikov\(^1\)* \quad Janis Keuper\(^{1,2}\)*

\(^1\)IMLA, Offenburg University, \(^2\)Fraunhofer ITWM

\{paul.gavrikov, janis.keuper\}@hs-offenburg.de

Abstract

It is common practice to apply padding prior to convolution operations to preserve the resolution of feature-maps in Convolutional Neural Networks (CNN). While many alternatives exist, this is often achieved by adding a border of zeros around the inputs. In this work, we show that adversarial attacks often result in perturbation anomalies at the image boundaries, which are the areas where padding is used. Consequently, we aim to provide an analysis of the interplay between padding and adversarial attacks and seek an answer to the question of how different padding modes (or their absence) affect adversarial robustness in various scenarios.

1. Introduction

Over the recent years, Convolutional Neural Networks (CNN) [1] have become the dominant backbone of most learning-based approaches for computer vision applications [2]. However, despite their overwhelming success in terms of achieving high test accuracies on various (vision) benchmarks, CNNs also have shown to be very vulnerable against minor changes in the input data distribution, e.g., against adversarial attacks [3]. This is especially concerning in safety-critical applications such as autonomous driving [4] or medical imaging [5] where human lives are at stake. Beyond adversarial training [6], which can be considered as the current standard approach towards the training of more robust networks, recent works showed that inherent signal processing flaws related to convolutions are one of several possible sources for the lack of robustness. While prior investigations, including down-sampling [7, 8], the size of the convolution kernels [9, 10] and the choice of the non-linear activation functions [11, 12], have shown a significant impact of these network components on the robustness of trained CNN models, there is no systematic analysis of the commonly applied padding schemes in this context. Our contribution is motivated by an initial analysis of spatial

*Funded by the German Ministry for Science, Research and Arts, Baden-Württemberg under Grant 32-7545.204/451 (Q-AMeLiA).
2. Related Work

Adversarial robustness. Neural networks tend to overfit the training data distribution and fail to generalize beyond. As such the predictions are often highly sensitive to small input perturbations [15, 16] that are (almost) imperceptible and semantically meaningless to humans. In some cases, these perturbations can be as small as a single pixel [17]. Formally, this phenomenon can be introduced as follows. Given a model $\mathcal{F}$ parameterized by $\theta$, an input sample $x$ with the corresponding class label $y$, and a loss function $\mathcal{L}$, an adversarial attack will attempt to maximize the loss $\mathcal{L}$ by finding an additive perturbation to $x$ in the $B_r(x)$ ball that is centered at $x$. The $\ell_p$-norm denoted by $\| \cdot \|_p$ is bounded by the radius (budget) $\epsilon$ to restrict perturbations to minor changes.

$$\max_{x' \in B_r(x)} \mathcal{L}(\mathcal{F}(x'; \theta), y)$$

$$B_r(x) = \{ x' : \| x - x' \|_p \leq \epsilon \}$$ (1)

Adversarial attacks can be found in both, white- and black-box settings [18, 19, 20, 21]. Amongst the most effective attacks are gradient-based white-box attacks that use the model prediction to perturb images in the direction of the highest loss [22, 6, 14, 23]. Models trained without adversarial defenses can typically not withstand attacks with high $\epsilon$ budgets. An unequivocal solution to overcome this phenomenon is adversarial training (AT) [6]. Adversarial training trains the model on worst-case perturbations found during training and effectively turns out-of-domain attacks to in-domain samples. As a side-effect, this results in models that classify based on shapes and not texture information which is better aligned with human vision [24, 25]. Further, they are less over-confident [26] than normally trained ones. Unfortunately, adversarial training is also susceptible to overfitting to the attacks employed during its training phase [27, 28, 29]. Hence, a common choice to assess robustness is AutoAttack [14] which compares against multiple attacks [30] such as APGD-CE [14], FAB [23], and Square [21].

Correlation between architecture and robustness. Tang et al. [31] performed a mass evaluation of different networks against robustness. They concluded that architecture plays an important role in robustness but no universal training recipe exists. For example, they recommend training light-weight architectures with AdamW [32], but SGD [33] performs significantly better for heavier architectures. Further, they state that CNNs outperform transformers on natural and system noise, but the opposite holds for adversarial robustness. Huang et al. [34] tested the influence of topology, kernel size, activation, normalization, and network size of residual networks on robustness. They find that under the same FLOPs, narrow-deep networks are more robust than shallow-wide networks and derive an optimal ratio. Further, they observe that pre-activation increases robustness, but increasing kernel size does not. Robust architectures can also be an optimization goal in Neural Architecture Search [35]. Gavrikov et al. [36, 37] provided an analysis in weight space and showed that convolution filters of adversarially trained CNNs are more diverse in learned patterns and generally occupy more of the network capacity than normally trained counterparts.

Padding. Prior to convolution, inputs can be artificially enlarged (padding), e.g., to maintain the resolution before and after convolutions. Padding is controlled by two parameters: the padding mode which determines the information in the border, and the padding size which determines the size of the border. Popular frameworks like PyTorch provide a multitude of padding modes (see Fig. 2 for examples):

- **zeros**: the padding area is filled with zeros.
- **reflect**: mirrors the input at the boundary.
- **replicate**: copies the last pixel at the image boundary into the padding area.
- **circular**: wraps the input around the boundary and continues at the opposite boundary(s).

Despite the variety in choice, few recent works have studied the importance of padding in convolution layers, and the majority of image classification networks after AlexNet [38] use zero padding. In theory, the padding size can be set independently on all image axes, most commonly though, it is set similar constant and determined by the kernel size $k$: $[k/2]$ (also called same padding). For instance, for $k = 3$ the padding size for same padding is 1. Multiple works showed that zero padding increases translational invariance [39, 40, 41] and circular padding appears to break it [39]. However, the commonly used same zero padding can cause the model to produce artifacts in feature-maps that result in the loss of visual features (blind spots) [42]. The authors link this to even input resolutions and show that switching to uneven resolutions improves accuracy. Although all

![Image](image-url)
other padding modes except zero padding reduce artifacts in feature-maps, the authors cannot recommend a single best padding mode, as the best choice appears to depend on the specific problem [43, 44, 45].

To the best of our knowledge, no investigation of the role of padding on robustness has been performed so far. We aim to close this gap in this work.

3. Experiments

For our experiments, we train models and switch the padding mode in all convolution layers between zeros, reflect, replicate, and circular. Further, we switch the convolution kernel size \( k \in \{3, 5, 7, 9\} \) and set the padding size to \( \lfloor k/2 \rfloor \) per side, accordingly (same padding). The trained models are attacked with APGD-CE [14], FAB [23], and Square [21] using the implementation in [14]. The attacks on models are performed under two different budgets, derived from trends in recent literature. For low budget attacks we generate attacks from \( p = 2, \epsilon = 0.1 \) and \( p = \infty, \epsilon = 1/255 \); and for high budget attacks \( p = 2, \epsilon = 0.5 \) and \( p = \infty, \epsilon = 8/255 \). Attacks are evaluated on a subset of 1,000 CIFAR-10 test samples. We measure the clean performance and robust performance under individual attacks and store and analyze perturbed inputs that have successfully fooled the model.

Training Details. We train 20-layer deep ResNets optimized for CIFAR-10 [13] (ResNet-20) as introduced in [46] models on CIFAR-10 [13] with the default train/validation splits. Training images are randomly horizontally flipped during training. Test images are not modified. Both splits are normalized by the channel mean and standard deviation. Training is executed with an SGD [33] optimizer (with Nesterov momentum [47] of 0.9) for 75 epochs, with an initial learning rate of 0.01 following a cosine annealing schedule [48], a weight decay [49] of 0.01, a batch size of 256, and cross-entropy loss [50] with a label smoothing [51] value of 0.1. For our analyses, we use the model checkpoints at the end of training. For adversarial training experiments, we follow the same parameters except that we train with an FGSM [22] \( p = \infty, \epsilon = 8/255 \) adversary and select the checkpoint with the highest accuracy against a PGD [52] \( p = \infty, \epsilon = 8/255 \) adversary (early stopping) [22] on the test data to prevent robust overfitting [53, 27, 28]. Although neither our training nor tested architecture is comparable with SOTA approaches (e.g., [54, 55, 56, 57, 34]), we believe it to be sufficiently well suited for this analysis. Unless stated otherwise we report results over 10 model runs with different randomness seeds.

3.1. Clean Performance

On clean data and without adversarial training, we see that zero padding outperforms all other padding modes in clean accuracy (Table 1). However, under adversarial training replicate or reflect outperform zero padding, depending on the kernel size. While the difference is not very high with larger kernels, at the common kernel size \( k = 3 \) it amounts to 1.27%. By a large margin, circular padding performs the worst in combination with adversarial training at all kernel sizes. We also observe that the accuracy falls off with increasing kernel size without adversarial training, while there is hardly a correlation on adversarially-trained models. However, it is worth noting that \( k > 3 \) models appear to reach higher accuracy under adversarial training.

<table>
<thead>
<tr>
<th>AT</th>
<th>( k )</th>
<th>Clean Test Performance [%] (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>zeros</td>
</tr>
<tr>
<td>×</td>
<td>3</td>
<td>90.26</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>90.14</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>89.36</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>88.22</td>
</tr>
<tr>
<td>✓</td>
<td>3</td>
<td>71.84</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>73.72</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>73.86</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>73.51</td>
</tr>
</tbody>
</table>

Table 1: Clean test accuracy of ResNet-20 trained on CIFAR-10 with different padding modes, kernel sizes, and adversarial training. Mean over 10 runs. Best, second best.

<table>
<thead>
<tr>
<th>AT</th>
<th>( k )</th>
<th>AutoAttack Performance [%] (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>zeros</td>
</tr>
<tr>
<td>×</td>
<td>3</td>
<td>8.52</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>17.69</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>29.06</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>39.18</td>
</tr>
<tr>
<td>✓</td>
<td>3</td>
<td>36.88</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>37.48</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>37.42</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>37.49</td>
</tr>
</tbody>
</table>

Table 2: AutoAttack [14] robust test accuracy of ResNet-20 trained on CIFAR-10 with different padding modes, kernel sizes, and adversarial training. Mean over 10 runs. Best, second best.

3.2. Robust Performance

Next, we compare the robust accuracy under different attacks while testing different padding approaches and kernel sizes (Fig. 3).
Native Robustness. First, we analyze the performance of models trained without adversarial defenses such as adversarial training. For low budgets, we observe that zeros outperforms all other padding modes in most settings by a large margin (Fig. 3a). Generally, we see a very similar trend for padding modes independent of the individual attack or norm, of course, except for the actual robust performance (FAB is the strongest attack, and Square is the weakest). Most notably, we observe that increasing $k$ seems to diminish the gap between zero padding and other modes. Interestingly, contrary to the observations of [34] we find that increasing kernel size also significantly improves robustness in this setting. For example, we see an improvement by 28.64% between $k = 3$ and $k = 9$ kernels against $\ell_\infty$-bounded APGD-CE adversaries. As expected, when switching to high-budget attacks (not pictured for brevity), all tested padding mode/kernel size combinations collapse below random performance and mostly even to near 0 performance. For $k = 3$, circular seems to be the only interesting outlier, as it gains a few percent in robust accuracy, yet, still underperforms a random baseline.

Adversarial Training. Additionally to native robustness, we aim to understand whether differences arise in models trained with adversarial training (Fig. 3b). Contrary to previous results, observing patterns in this setting becomes less straightforward. Averaging over all attacks, we again see that robustness improves with increasing kernel size, albeit at an almost insignificant rate for $k \geq 5$. Additionally, all padding modes seem to perform reasonably well but depending on the attack, norm, or kernel size individual modes perform better than others. The only exception to this is circular padding. Although it performed reasonably well regarding native robustness, in combination with adversarial training it is always by far the worst choice. Regarding the other padding modes, we make the following observations: If we average the robust performance over all attacks and norms we see the same patterns as for clean accuracy, reflect is the best choice for $k = 3, 9$, replicate performs best for $k = 5, 7$. However, we also see large fluctuations. For example, for $k = 5$ reflect becomes the worst choice after circular with replicate being the best, on $k = 7$ the difference between reflect and replicate is marginal as both perform almost equally well and, finally, at $k = 9$ we see a switch and replicate becomes the worst choice after circular while reflect becomes the best choice. Interestingly, zero padding is never the best option on average over all attacks. Yet, if we only analyze $\ell_\infty$-bounded APGD-CE attacks, it outperforms all other padding modes by a non-negligible margin. This is a concern when evaluating with AutoAttack [14] which starts with APGD-CE attacks, and proceeds with other attacks only on samples that could not be successfully attacked. As such it is not surprising that in terms of AutoAttack accuracy, zero padding remains the best-performing method Tab. 2. However, replicate/reflect padding modes are usually only marginally worse and come at the benefit of improved clean accuracy.

3.3. Padding Anomalies

Following our robustness performance analysis, we now aim to understand where perturbations are primarily located and how different padding modes influence their distribu-
tion on $k = 3$ models. To this end, we average the absolute differences between successful adversarial examples (i.e. those that flip the predicted label) and the corresponding clean samples (Fig. 1). We separate between normally-trained (native) models and adversarially-trained models, but this time evaluate both under high budgets to increase perturbations aiming to improve visibility. For a better comparison between tested dimensions, we project perturbation magnitudes to the image X-axis (Fig. 4). Generally, though, we see relatively similar observations on the Y-axis, except that perturbations are less symmetrically distributed and increase towards the lower image edge.

**Native Models.** We see significant differences (Fig. 4a) between $\ell_2$-perturbations which are primarily located in the center of the image and $\ell_\infty$-perturbations which appear to be distributed more uniformly across the image but contain anomalies in outermost pixels or their close neighbors. Noticeably, these are areas where the receptive field intersects with the padded area. For $\ell_\infty$-bounded APGD-CE, and to a lesser extent also FAB, we see increases in perturbations at boundaries, while Square shows strong decreases in perturbations at boundaries. Generally, we see similar distribution shapes independent of padding mode, except for $\ell_2$-bounded APGD-CE where reflect/replicate show increased perturbations at the boundaries. However, we see a different area under the curve (AUC) for padding modes and zero shows the largest AUC.

**Adversarial Training.** Under adversarial training (Fig. 4b) perturbation distributions of $\ell_\infty$-bounded attack-
and \textit{circular} introduces an overhead of approximately 137\% - presumably, due to optimized implementations. Our results show that the superiority of \textit{replicate}, \textit{reflect} in some settings comes at the cost of slower test and thus also train time.

### 3.5. Effect on Model Decisions

Our previous results showed that the choice of padding method and size can have significant effects on both, the model robustness (Tab. 1 and Fig. 3) and the placement of the perturbations (Fig. 1). To further analyze whether the perturbation anomalies at image edges are affecting the model decision intrinsics, and thus the robustness, we analyze visual explanations via \textit{LayerCAM} [58] as implemented in [59]. Explanations are computed for the feature-maps after the last residual block with respect to the predicted label. We compute the differences in explanations between an adversarial example and its clean counterpart whenever the attack was successful and visualize the mean over all samples of 10 differently seeded models (Fig. 5). For this analysis, we limit ourselves to \( \ell_\infty \)-bounded high-budget APGD-CE attacks on \( k = 3 \) models. There is a clear shift in explanations between normally trained and adversarially-trained models. For normally trained models under low-budget attacks (Fig. 5a) we observe a difference in the explanation shift depending on the padding mode. With zero padding, explanations clearly shift toward the center indicating that the attack actually concentrates on the image foreground (as CIFAR-10 is well-centered). Under \textit{circular} padding the shift area seems to stretch horizontally and attacks start to aim at image boundaries. \textit{Replicate} and \textit{reflect} padding modes show almost similar behavior: while there is some shift towards the center, there are increasingly more attacks on the image boundary (except for the top left corner) than with previous modes. As we increase the attack budget (Fig. 5b) we observe the same behavior for all padding modes except zero padding. Attacks move entirely to the image boundaries padding showing that the adversary attacks everything except the image foreground. The shift intensity is smaller for \textit{circular} than for \textit{replicate/reflect}. For zero padding we observe hardly any differences to the low-budget setting. For adversarially-trained models (Fig. 5c) we see quite the opposite: attacks for all padding modes are shifted toward the image edges, with no clear difference between padding modes except for the increased intensity of shifts in \textit{circular} models. Although not pictured here for brevity, we observe similar trends for \( \ell_2 \)-bounded attacks.

### 3.6. Is No Padding a Better Alternative?

In this next section, we aim to understand the role of padding per se on robustness by entirely removing padding from all convolution layers, and without padding, models should not develop boundary anomalies. Without any other changes, this would result in smaller feature-map (representation) resolutions, and, residual networks would experience a size mismatch between input signals propagated between skip connections and the actual convolution paths. To solve the size mismatch after skip connections we zero-pad the processed signal before the summation instead of directly summing the residual and processed signal. Note that this is different from zero-padded convolutions as it does not result in zero-padded inputs to the next operators. Regarding the smaller feature-maps, we compare \( k = 3 \) padding-free networks in 3 settings:

- **Unmodified (None):** We use the same 32 \( \times \) 32 px CIFAR-10 samples as before. For ResNet-20 this results in 4 \( \times \) 4 representations at the deepest layers instead of 8 \( \times \) 8.

### Table 3: Benchmark of the average time for padding or padded 2D convolution operation under different padding modes. Bold marks best.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Average time for operation [( \mu s )] (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>zeros</td>
</tr>
<tr>
<td>Only padding</td>
<td>21.87</td>
</tr>
<tr>
<td>2D Convolution</td>
<td><strong>55.65</strong></td>
</tr>
</tbody>
</table>

![Figure 5: Average shifts in LayerCAM [58] explanations following successful \( \ell_\infty \)-bounded APGD-CE attacks under different padding modes. Mean computed over 10 models. Orange areas indicate areas where explanations move under adversarial attacks, while blue areas indicate areas of explanations have been removed.](image)
• Upscaling (None + Up): We upscale CIFAR-10 samples to 48 × 48 px via bilinear interpolation [60] to match the representations of padded networks.

• Outpainting (None + Out): To increase the resolution in a meaningful manner we experiment with image outpainting to 48 × 48 px via MAT [61] pre-trained on the Places dataset [62] containing scenic images that should not interfere with the original label (Fig. 6). We outpaint both, train and test samples.

Without adversarial training, we see a drop in clean and robust accuracy if we simply disable padding without countermeasures (Tab. 4). With both, upscaling or outpainting, we are able to mitigate the drop in clean accuracy, yet provoke a complete failure in robust accuracy except for Square attacks but still at a decreased robustness. Although it is worth noting that \( \ell_\infty \)-bounded attacks can attack more area for the same budget when increasing the image resolution as we do in upscaling or outpainting, the delta is too large to be caused by only an increased attack area. Under adversarial training, we see similar trends, but this time we also measure a clear difference between upscaling and outpainting. Surprisingly, outpainting seems even further to reduce robustness. However, we also notice that outpainting seems to affect non-masked areas (the original sample) and tampers with image statistics. In all cases, disabling padding results in worse performance and is therefore not recommended.

4. Conclusion & Discussion

We have evaluated the adversarial robustness of CIFAR-10 models under different padding modes, kernel sizes, attacks, and training modes. Based on our results, we can provide the following recommendations:

For settings targeting native robustness, we advise using zero padding and increasing the kernel/padding size, if possible. We hypothesize that larger kernels are more likely to cause robustness than larger padding sizes. Switching to other padding modes clearly deteriorates the performance and starts shifting attacks toward image boundaries.

When using adversarial training, it can be fruitful to experiment with the reflect and replicate padding modes. However, commonly models are benchmarked with AutoAttack [14] which starts with APGD-CE attacks where zero padding performs best, and proceeds with other attacks only on samples that could not be successfully attacked. Since APGD-CE is usually a very effective attack, the difference measured based on the remaining subset may not be significant to show clear differences between padding modes. However, at negligible impairments of robustness performance, we saw an improved clean accuracy of non-zero padding modes. In addition, we suggest reporting adversarial robustness for multiple attacks separately when studying padding.

Lastly, in all cases, we can recommend the usage of padding as padding-free architectures performed significantly worse in all investigated scenarios.

Limitations. For completeness, we want to emphasize that we only experimented on CIFAR-10. As with many “toy-datasets”, objects in question are usually perfectly centered in the images. Unfortunately, this applies to most common benchmarking datasets such as MNIST [63], SVHN [64], CIFAR-100 [13], ImageNet [65] etc. As such, we are not confident that we would obtain significantly different results on these datasets. Repeating our experiments on less curated datasets may, however, result in different observations. Further, we only experimented with ResNet-20. While we believe that the results are representative of the popular residual networks in general, we cannot guarantee that our results scale to entirely different architectures. We aim to bridge these gaps in future work.

References


Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, eds.), 2022. 2


[34] Shihua Huang, Zhichao Lu, Kalyanmoy Deb, and Vishnu Naresh Boddeti, “Revisiting residual networks for adversarial robustness: An architectural perspective,” 2022. 2, 3, 4


[57] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli, “Uncovering the limits of adversarial training against norm-bounded adversarial examples,” 2020. 3


[64] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bis- sacco, Bo Wu, and Andrew Ng, “Reading digits in natural images with unsupervised feature learning,” *NIPS*, 01 2011. 7