

# FrequencyLowCut Pooling - Plug & Play against Catastrophic Overfitting

Julia Grabinski<sup>1,2</sup>, Steffen Jung<sup>4</sup>, Janis Keuper<sup>2,3</sup>, and Margret Keuper<sup>1,4</sup>

<sup>1</sup> University of Siegen, Germany

<sup>2</sup> CC-HPC, Fraunhofer ITWM, Kaiserslautern, Germany

<sup>3</sup> Institute for Machine Learning and Analytics, Offenburg University, Germany

<sup>4</sup> Max Planck Institute for Informatics, Saarland Informatics Campus, Germany

**Abstract.** Over the last years, Convolutional Neural Networks (CNNs) have been the dominating neural architecture in a wide range of computer vision tasks. From an image and signal processing point of view, this success might be a bit surprising as the inherent spatial pyramid design of most CNNs is apparently violating basic signal processing laws, i.e. *Sampling Theorem* in their down-sampling operations. However, since poor sampling appeared not to affect model accuracy, this issue has been broadly neglected until model robustness started to receive more attention. Recent work [17] in the context of adversarial attacks and distribution shifts, showed after all, that there is a strong correlation between the vulnerability of CNNs and aliasing artifacts induced by poor down-sampling operations. This paper builds on these findings and introduces an aliasing free down-sampling operation which can easily be plugged into any CNN architecture: FrequencyLowCut pooling. Our experiments show, that in combination with simple and fast FGSM adversarial training, our hyper-parameter free operator significantly improves model robustness and avoids catastrophic overfitting.

**Keywords:** CNNs, Adversarial Robustness, Aliasing

## 1 Introduction

The robustness of convolutional neural networks has evolved to being one of the most crucial computer vision research topics in recent years. While state-of-the-art models provide high accuracies in many tasks, their susceptibility to adversarial attacks [7] and even common corruptions [19] is hampering their deployment in many practical applications. Therefore, a wide range of publications aims at providing models with increased robustness by adversarial training schemes [14,41,46], sophisticated data augmentation techniques [34] and enriching the training with additional data [4,15]. As a result, robust models can be learned with common CNN architectures, yet arguably at a high training cost - even without investigating the reasons for CNN's vulnerability. These reasons are of course multi-fold, starting with the high dimensionality of the feature

space and sparse training data such that models easily tend to overfit [35,43]. Recently, the pooling operation in CNNs has been discussed in a similar context for example in [17] who measured the correlation between aliasing and a network’s susceptibility to adversarial attacks. [48] have shown that commonly used pooling operations even prevent the smoothness of image representations under small input translations. In this paper, we propose a simple and straight-forward way to prevent aliasing during the down-sampling operation in convolutional neural networks. Therefore, we propose to directly implement down-sampling as a cropping operation in the frequency domain, which we refer to as FrequencyLowCut pooling. In contrast to previously proposed blurring operations [48,49], this procedure guarantees that frequencies beyond the Nyquist rate are ignored in the classification process and the resulting model can not overfit to high-frequency details. Natively, resulting models exhibit only slightly improved robustness against adversarial attacks and common corruptions - as issues regarding the high dimensionality of the feature space persist. Yet, when paired with simple and efficient single step adversarial training [14], this approach allows to learn models on par with recent, highly optimized adversarial training schemes [4,34,44]. Previous approaches based on single step attacks suffer from catastrophic overfitting and therefore require early stopping criteria [35,43]. Our proposed model outperforms such approaches as well as previous approaches addressing the down-sampling smoothness through spatial blurring[48,49] by a significant margin.

Our contributions are summarized as follows:

- We introduce FrequencyLowCut pooling (FLC), ensuring aliasing-free down-sampling within CNNs.
- Through extensive experiments with various datasets and architectures, we show empirically that FLC pooling prevents single step adversarial training from catastrophic overfitting, while this is not the case for other recently published improved pooling operations (e.g. [48]).
- FLC pooling is significantly faster, around five times, and easier to integrate than previous adversarial training or defence methods. It provides a hyperparameter free plug and play module for increased model robustness.

## 1.1 Related Work

**Adversarial Attacks.** While CNNs are known for their excellent performance on image classification tasks, they are susceptible to adversarial attacks [14,32,40], i.e. to intentional image perturbations. Recently, many different adversarial attacks as well as defences have been developed. One of the earliest attacks is the Fast Gradient Sign Method (FGSM) by [14], followed by more sophisticated methods like Projected Gradient Descent (PGD) [26], DeepFool (DF) [32], Carlini and Wagner (CW) [3] or Decoupling Direction and Norm (DDN) [36]. While single step adversarial examples, like FGSM, take the full possible perturbation step in the range of  $\epsilon$  in one step, PGD iteratively searches for the minimal perturbation size needed to flip the inferred labels of the attacked samples over

a given number of iterations. Typically,  $\epsilon$  is either measured in  $L_2$  or  $L_{\text{inf}}$  norms and given values are often expressed as fractions of eight-bit encodings of input color channels, like  $\epsilon = \frac{1}{255}$ . The popular choice of  $\epsilon = \frac{8}{255}$  is motivated by the limited color resolution of the human eye [13]. Recently, AutoAttack [7], an ensemble of different attacks including an adaptive version of PGD, has been established as the current baseline for adversarial robustness and is used to evaluate the robustness benchmarks in RobustBench [6].

In relation to image down-sampling, [45] and [29] demonstrated steganography based attacks on the pre-processing pipeline of CNNs.

**Adversarial Training.** Most proposed attacks in literature already come with a dedicated defence to counter their adversarial examples [14,36]. Beyond these attack-specific defences, there are many methods for more general adversarial training (AT) schemes. These typically either add loss terms to be more robust against a special type of adversarial noise [11,46] or introduce additional training data [4,38]. Some approaches also combine both [41] or use data augmentation [15], which is typically combined with weight averaging techniques [34]. A widely used source for additional training data is *ddpm* [16,33,34], which contains one million extra samples for CIFAR-10. This dataset is generated with the model proposed by [20]. RobustBench [6] gives an overview and evaluation of a variety of models w.r.t. their adversarial robustness and the additional data used. [16] receive an additional boost in robustness by adding specifically generated images while [33] add wrongly labeled data to the training-set.

A common drawback of all adversarial training methods is the vast increase in computation needed to train networks: large amounts of additional adversarial samples and slower convergence due to the harder learning problem typically increase the training time by a factor between seven and fifteen [26,41,44,46].

**Catastrophic Overfitting.** Adversarial training with single step FGSM is a simple approach to achieve basic adversarial robustness [5,35]. Unfortunately, the robustness of this approach against stronger attacks like PGD is starting to drop again after a certain amount of training epochs. [43] called this phenomenon *catastrophic overfitting*. They concluded, that one step adversarial attacks tend to overfit the chosen adversarial perturbation magnitude (given by  $\epsilon$ ) but fail to be robust against multi-step attacks like PGD. [35] introduced early stopping as counter measure. After each training epoch, the model is evaluated on a small portion of the dataset with a multi-step attack, which again increases the computation time. As soon as the accuracy drops compared with a hand selected threshold the model training is stopped. [24] and [39] showed that the observed overfitting is related to the flatness of the loss landscape. They introduced a method to compute the *optimal* perturbation length  $\epsilon'$  for each image and do single step FGSM training with this optimal perturbation length to prevent catastrophic overfitting. [2] showed that catastrophic overfitting not only occurs in deep neural networks but can also be present in single-layer convolutional neural networks. They propose a new kind of regularization, called GradAlign

to improve FGSM perturbations and flatten the loss landscape to prevent catastrophic overfitting.

**Anti-Aliasing.** The problem of aliasing effects in the context of CNN based neural networks has already been addressed from various angles in literature: [48] established Anti-Aliasing methods in convolutional filters in order to improve the shift-invariance of CNN classifiers. This approach has been further improved by [49] by using learned instead of predefined blurring filters. [28] use the low frequency components of wavelets to suppress the effects of aliasing in order to increase the robustness of pooling operations against common image corruptions. Here, we show that aliasing is not only relevant for robustness to common corruptions but also affects adversarial robustness. [21] propose not only a depth adaptive blurring filter before pooling but also an anti-aliasing activation function. This activation function is inspired by C-ReLu but uses a smooth roll-off phase instead of the sharp cutoff at threshold  $t$ . Also, [23] achieve aliasing free generators for GANs by blurring before sampling and non-linearities, like ReLu, whereas [10] and [22] address aliasing in GAN generators by employing additional loss terms in the frequency space. Lately, [17] showed that adversarially robust models exhibit much less aliasing in their down-sampling layers than non-robust models.

## 2 Preliminaries

### 2.1 Adversarial Training

In general, adversarial training can be formalized as an optimization problem given by a *min-max* formulation:

$$\min_{\theta} \max_{\delta \in \Delta} L(x + \delta, y; \theta), \quad (1)$$

where we seek to optimize network weights  $\theta$  such that they minimize the loss  $L$  between inputs  $x$  and labels  $y$  under attacks  $\delta$ . The maximization over  $\delta$  can thereby be efficiently performed using the Fast Gradient Sign Method (**FGSM**), which takes one big step defined by  $\epsilon$  into the direction of the gradient [14]:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y)). \quad (2)$$

The Projected Gradient Descent method, (**PGD**), works similar to FGSM but instead of taking one big step in the direction of the gradient with step size  $\epsilon$ , it iteratively optimizes the adversarial example with a smaller, defined step size  $\alpha$ . Random restarts further increase its effectiveness. The final attack is clipped to the maximal step size of  $\epsilon$ .

$$x'_{N+1} = \text{Clip}_{X, \epsilon} \{x'_N + \alpha \cdot \text{sign}(\nabla_x L(\theta, x, y))\} \quad (3)$$

PGD is one of the strongest attacks, due to its variability in step size and its several random restarts. Yet, its applicability for adversarial training is limited

as it requires a relatively long optimization time for every example. Additionally, PGD is dependent on even several hyperparameters in practice which makes it even less attractive for training. In contrast, FGSM is fast and straight-forward to implement. Yet, models that use FGSM for adversarial training tend to overfit on FGSM attacks and are not robust to other attacks such as PGD, i.e. they suffer from catastrophic overfitting [43].

## 2.2 Down-sampling in CNNs

Independent of their actual network topology, CNNs essentially perform a series of stacked convolutions and non-linearities. Using a vast amount of learnable convolution filters, CNNs are capable to extract local texture information from all intermediate representations (input data and feature-maps). In order to be able to abstract from this localized spatial information and to learn higher order relations of parts, objects and entire scenes, CNNs apply down-sampling operations to implement a spatial pyramid representation over the network layers.

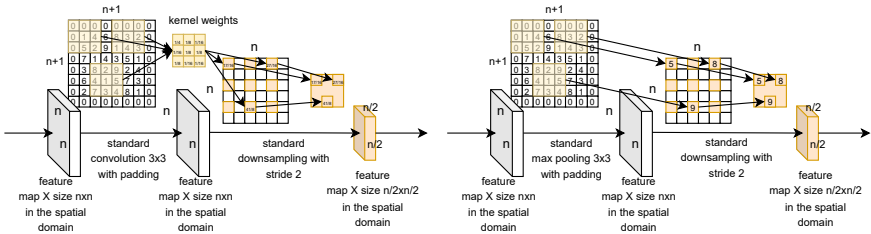


Fig. 1: Standard down-sampling operations used in CNNs. Left: down-sampling via convolution with stride two. First the featuremap is padded and the actual convolution is executed. The stride defines the step-size of the kernel. Hence, for stride two, the kernel is moved two spatial units. In practice, this down-sampling is often implemented by a standard convolution with stride one and then discarding every second point in every spatial dimension. Right: down-sampling via MaxPooling. Here the max value for each spatial window location is chosen and the striding is implemented accordingly.

This down-sampling is typically performed via a convolution with stride greater than one or by so-called pooling layers (see Fig. 1). The most common pooling layers are AveragePooling and MaxPooling. All of these operations are highly sensitive to small shifts or noise in the layer input.

**Aliasing.** CNNs usually have a pyramidal structure in which the data is progressively sub-sampled in order to aggregate spatial information while the number of channels increases. During sub-sampling, no explicit precautions are taken to

avoid aliases, which arise from under-sampling. Specifically, when sub-sampling with stride two, any frequency larger than  $N/2$ , where  $N$  is the size of the original data, will cause pathological overlaps in the frequency spectra (Fig. 2). Those overlaps in the frequency spectra cause ambiguities such that high frequency components appear as low frequency components. Hence, local image perturbations (noise) can become indistinguishable from global manipulations. [17] showed that aliasing in CNNs strongly coincides with the robustness of the model. Based on this finding, one can hypothesise that models that overfit high frequencies in the data tend to be less robust. This thought is also in line with the widely discussed texture bias [12]. To substantiate this hypothesis in the context of adversarial robustness, we investigate and empirically show that catastrophic overfitting coincides with an increase in aliasing during adversarial training. Based on this observation, we expect networks that sample without aliasing to be better behaved in adversarial training settings. The Frequency-LowCut pooling, which we propose in the following, trivially fulfills this property.

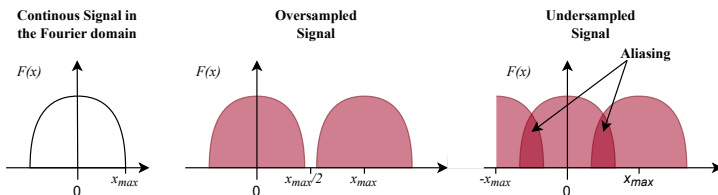


Fig. 2: Aliasing in the frequency domain. Left a 1D signal with the  $x_{\max}$  as maximal frequency is shown in the frequency domain. When this signal is down-sampled, the original signal is replicated at a distance depending on the sampling rate. If sampled at a sufficiently large sampling rate, the distance between the replica is large and the signals do not overlap (middle). If the sampling rate is too low, the signal is under-sampled and aliases are observed due to overlapping replica (right).

### 3 FrequencyLowCut Pooling

Several previous approaches such as [48,49] reduce high frequencies in features maps before pooling to avoid aliasing artifacts. They do so by classical blurring operations in the spatial domain. While those methods reduce aliasing, they can not entirely remove it due to sampling theoretic considerations in theory and limited filter sizes in practice (see supplementary material or [13] for details). We aim to perfectly remove aliases in CNNs down-sampling operations without adding additional hyperparameters. Therefore, we directly address the down-sampling operation in the frequency domain, where we can sample according to the Nyquist rate, i.e. without any aliases, in a straight-forward way. In practice, the proposed down-sampling operation first performs a Fast Fourier Transform

(FFT) of the feature maps  $f$ . Feature maps with height  $M$  and width  $N$  to be down-sampled are then represented as

$$F(k, l) = \frac{1}{NM} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-2\pi j \left( \frac{k}{M} m + \frac{l}{N} n \right)}. \quad (4)$$

In the resulting frequency space representation  $F$ , all frequencies larger than the sampling rate over two have to be discarded before down-sampling to avoid aliases. Sampling of the feature maps in this space amounts to cropping exactly the remaining low frequencies from  $F$ . Since down-sampling in CNNs is commonly done with a sampling rate of two, we will examine our FCL pooling with this sampling rate.

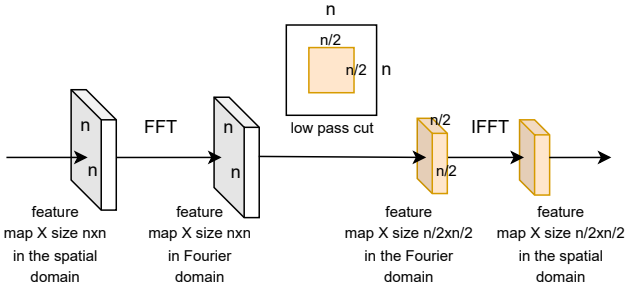


Fig. 3: FrequencyLowCut pooling, the proposed, guaranteed alias-free pooling operation. We first transform feature maps into frequency space via FFT, then crop the low frequency components. The result is transformed back into the spatial domain. It corresponds to a sinc-filtered and downsampled feature map and is fed into the next convolutional layer.

Then we shift the low frequency components into the middle of the  $k$ -space and obtain  $F_s$ . Afterwards, we crop the low frequency components below the Nyquist frequency  $F_{sd}$  by selecting the frequency space center with  $M' = \frac{M}{4}$  and  $N' = \frac{N}{4}$ .

$$F_{sd} = F_s[:, :, M' : 3M', N' : 3N'] \quad (5)$$

After cropping, we shift the feature maps  $F_{sd}$  with height  $\hat{K}$  and width  $\hat{L}$  back to compute  $F_d$ . We transform  $F_d$  back to the spatial domain to achieve fully aliasing free down-sampled feature maps  $f_d$  with height  $\frac{M}{2}$  and width  $\frac{N}{2}$ .

$$f_d(\hat{m}, \hat{n}) = \frac{1}{\hat{K}\hat{L}} \sum_{k=0}^{\hat{K}-1} \sum_{l=0}^{\hat{L}-1} F_d(k, l) e^{2\pi j \left( \frac{\hat{m}}{\hat{K}} k + \frac{\hat{n}}{\hat{L}} l \right)}, \quad (6)$$

Fig. 3 shows the procedure in detail. In the spatial domain, the operation would amount to convolving the feature map with an infinitely large (non-bandlimited) sinc( $m$ ) =  $\frac{\sin(m)}{m}$  filter, which can not be implemented in practice.

Table 1: Clean training of PreAct-ResNet-18 architectures on CIFAR-10. We compare clean and robust accuracy against FGSM [14] with  $L_{\text{inf}}$ ,  $\epsilon = \frac{8}{255}$ , PGD [26] with  $L_{\text{inf}}$ ,  $\epsilon = \frac{1}{255}$  as well as  $L_2$  with  $\epsilon = 0.5$  (20 iterations) and common corruptions (CC) [19] (mean over all corruptions and severities).

Method	Clean	FGSM $\epsilon = \frac{8}{255}$	PGD $\epsilon = \frac{1}{255}$	$L_{\text{inf}}$ $\epsilon = 0.5$	PGD $L_2$ $\epsilon = 0.5$	CC
Baseline	<b>95.08</b>	34.08	7.15	6.68	74.38	
FLC Pooling	94.66	<b>34.65</b>	<b>10.00</b>	<b>11.27</b>	<b>74.70</b>	

## 4 Experiments

### 4.1 Native Robustness of FLC pooling

In the first experiment, we evaluate our proposed FLC pooling in a standard training scheme with PreAct-ResNet-18 architectures on CIFAR-10 (see appendix for details). Table 1 shows that both the decrease in clean accuracy as well as the increase in robustness are marginal compared to the baseline models. We argue that these results are in line with our hypothesis that the removal of aliasing artifacts alone will not lead to enhanced robustness and we need to combine correct down-sampling with adversarial training to compensate for the persisting problems induced by the very high dimensional decision spaces in CNNs.

### 4.2 FLC pooling for FGSM training

In the following series of experiments we apply simple FGSM adversarial training with  $\epsilon = \frac{8}{255}$  on different architectures and evaluate the resulting robustness with different pooling methods. We compare the models in terms of their clean, FGSM, PGD and AutoAttack accuracy, where the FGSM attack is run with  $\epsilon = 8/255$ , PGD with 50 iterations and 10 random restarts and  $\epsilon = 8/255$  and  $\alpha = 2/255$ . For AutoAttack, we evaluate the standard  $L_{\text{inf}}$  norm with  $\epsilon = 8/255$  and a smaller  $\epsilon$  of  $1/255$ , as AutoAttack is almost too strong to be imperceptible to humans [30]. Additionally, we evaluate AutoAttack with  $L_2$  norm and  $\epsilon = 0.5$ .

**CIFAR-10.** Table 2 shows the evaluation of a PreAct-ResNet-18 as well as a WideResNet-28-10 (WRN-28-10) on CIFAR-10 [25]. For both network architectures, we observe that our proposed FLC pooling is the only method that is able to prevent catastrophic overfitting. All other pooling methods heavily overfit on the FGSM training data, achieving high robustness towards FGSM attacks, but fail to generalize this towards PGD or AutoAttack. Our hyper-parameter free approach also outperforms early stopping methods which are additionally suffering from the difficulty that one has to manually choose a suitable threshold in order to maintain the best model robustness.



Table 2: FGSM adversarial training of Preact-ResNet-18 and WRN-28-10 architectures on CIFAR-10. Comparison of clean and robust accuracy (high is better) against PGD [26] and AutoAttack [7] on the full dataset with  $L_{\text{inf}}$  with  $\epsilon = 8/255$  and  $L_2$  with  $\epsilon = 0.5$ . FGSM test accuracies indicate catastrophic overfitting on the adversarial training data, hence this column is set to gray.

Method	Clean	FGSM	PGD	$L_{\text{inf}}$	AA	$L_{\text{inf}}$	AA	$L_2$	AA	$L_{\text{inf}}$
		$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = 0.5$	$\epsilon = 0.5$	$\epsilon = 0.5$	$\epsilon = 0.5$	$\epsilon = \frac{1}{255}$
<b>Preact-ResNet-18</b>										
Baseline: FGSM training	<b>90.81</b>	90.37	0.16	0.00	0.01	53.10				
Baseline & early stopping	82.88	61.71	11.82	3.76	17.44	72.95				
BlurPooling [48]	86.24	78.36	1.33	0.06	1.96	66.88				
Adaptive BlurPooling [49]	90.35	77.39	0.23	0.00	0.07	39.00				
Wavelet Pooling [27]	85.02	64.16	12.13	5.92	19.65	10.08				
FLC Pooling (ours)	84.81	58.25	<b>38.41</b>	<b>36.69</b>	<b>55.58</b>	<b>80.63</b>				
<b>WRN-28-10</b>										
Baseline: FGSM training	86.67	83.64	1.64	0.09	1.47	59.39				
Baseline & early stopping	82.29	56.36	31.26	28.54	46.03	76.87				
Blurpooling [48]	91.40	89.44	0.22	0.00	0.00	38.45				
Adaptive BlurPooling [49]	91.10	89.76	0.00	0.00	0.00	7.42				
Wavelet Pooling [27]	<b>92.19</b>	90.85	0.00	0.00	0.00	10.08				
FLC Pooling (ours)	84.93	53.81	<b>39.48</b>	<b>38.37</b>	<b>52.89</b>	<b>80.27</b>				

**CINIC-10.** Table 3 shows similar results on CINIC-10 [8]. Our model exhibits no catastrophic overfitting, while previous pooling methods do. It should be noted that CINIC-10 is not officially reported by AutoAttack. This might explain why the accuracies under AutoAttack are higher on CINIC-10 than on CIFAR-10. We assume that AutoAttack is optimized for CIFAR-10 and CIFAR-100 and therefore less strong on CINIC-10.

**CIFAR-100.** Table 4 shows the results on CIFAR-100 [25], using the same experimental setup as for CIFAR-10 in table 2. Due to the higher complexity of CIFAR-100, with ten times more classes than CIFAR-10, adversarial training tends to suffer from catastrophic overfitting much later (in terms of epochs) in the training process. Therefore we trained the Baseline model for 300 epochs. While the gap towards the robustness of other methods is decreasing with the amount of catastrophic overfitting, our method still outperforms other pooling approaches in most cases - especially on strong attacks.

**Analysis.** The presented experiments on several datasets and architectures show that baseline FGSM training, as well as other pooling methods, strongly overfit on the adversarial training data and do not generalize their robustness towards other attacks. We also showed that our FLC pooling sufficiently prevents

Table 3: FGSM adversarial training on CINIC-10 for PreAct-ResNet-18 architectures. We compare clean and robust accuracy (higher is better) against PGD [26] as well as AutoAttack [7] on the full dataset with  $L_{\text{inf}}$  with  $\epsilon = 8/255$  and  $L_2$  with  $\epsilon = 0.5$ . FGSM test accuracies indicate catastrophic overfitting on the adversarial training data, hence this column is set to gray.

Method	Clean	FGSM	PGD	$L_{\text{inf}}$	AA	$L_{\text{inf}}$	AA	$L_2$	AA	$L_{\text{inf}}$
		$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = 0.5$	$\epsilon = \frac{1}{255}$			
Baseline	87.46	58.83	1.31	0.12	1.55	55.21				
Baseline & early stopping	82.79	42.58	27.55	30.76	50.28	<b>79.88</b>				
Blurpooling [48]	87.13	54.16	1.29	0.20	4.68	70.56				
Adaptive BlurPooling [49]	<b>90.21</b>	52.27	0.05	0.00	0.01	40.96				
Wavelet Pooling [27]	88.81	64.16	1.76	0.12	3.38	66.61				
FLC Pooling (ours)	82.56	38.39	<b>36.28</b>	<b>49.61</b>	<b>60.51</b>	78.50				

Table 4: FGSM adversarial training on CIFAR-100 for PreAct-ResNet-18 architectures. We compare clean and robust accuracy (higher is better) against PGD [26] as well as AutoAttack [7] on the full dataset with  $L_{\text{inf}}$  with  $\epsilon = 8/255$  and  $L_2$  with  $\epsilon = 0.5$ . FGSM test accuracies indicate robustness to training data, hence this column is set to gray. Here, none of the models overfit, while FLC pooling still yields best overall robustness.

Method	Clean	FGSM	PGD	$L_{\text{inf}}$	AA	$L_{\text{inf}}$	AA	$L_2$	AA	$L_{\text{inf}}$
		$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = \frac{8}{255}$	$\epsilon = 0.5$	$\epsilon = \frac{1}{255}$			
Baseline	51.92	23.25	15.41	11.13	25.67	44.53				
Baseline & early stopping	52.09	23.34	15.51	10.88	25.78	44.61				
Blurpooling [48]	52.68	23.40	16.81	12.43	26.79	45.68				
Adaptive BlurPooling [49]	52.08	9.77	18.68	6.05	11.32	21.04				
Wavelet Pooling [27]	<b>55.08</b>	25.70	18.36	13.76	<b>27.51</b>	47.52				
FLC Pooling (ours)	54.66	26.82	<b>19.83</b>	<b>15.40</b>	26.30	<b>47.83</b>				

this catastrophic overfitting and is able to generalize robustness over different networks, datasets and attack sizes in terms of different  $\epsilon$ -values.

**Attack Structures** In Figure 4, we visualize AutoAttack adversarial attacks. Perturbations created for the baseline trained with FGSM differ significantly from those created for FLC pooling trained with FGSM. While perturbations for the baseline model exhibit high frequency structures, attacks to FLC pooling rather affect the global image structure.

### 4.3 Results on ImageNet

Additionally to the experiments on smaller datasets (like CIFAR and CINIC), we trained a network including FLC pooling on a larger dataset, namely Ima-

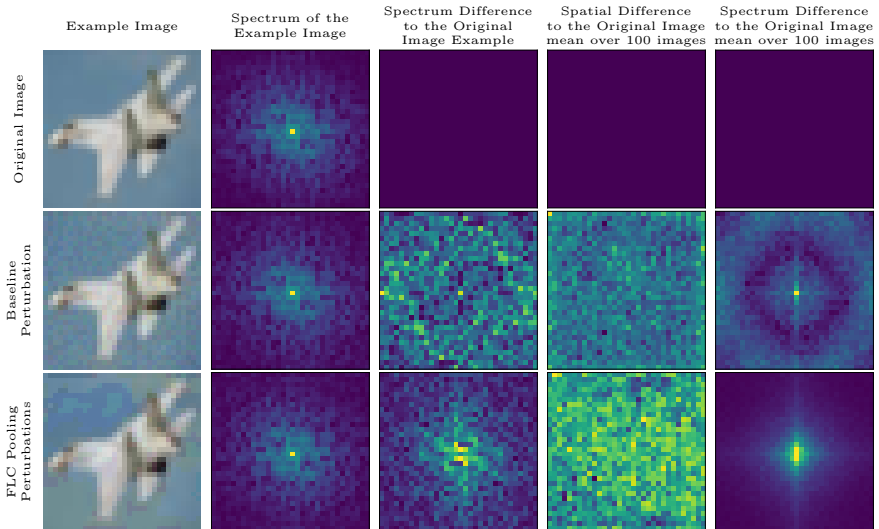


Fig. 4: Spatial and spectral differences of adversarial perturbations created by AutoAttack with  $\epsilon = \frac{8}{255}$  on the baseline model as well as our FLC Pooling. On the left side for one specific example of an airplane and on the right side the average difference over 100 images.

geNet [9]. Here we might not encounter the issue of robust overfitting for now, but most adversarial training methods require long training times to achieve adequate results on such a large dataset. We trained a ResNet50 architecture with FLC pooling on ImageNet on 4 NVIDIA A100 GPUs, resulting in 63.52% clean accuracy and 27.29% robust accuracy under auto attack. The median wall clock time per epoch is around 35 minutes. Table 5 shows our FLC Pooling in comparison to other approaches on ImageNet reported on RobustBench [6]. The training time of our model should be comparable to the one from Wong et al. [43], while other reported methods have significantly longer training times. Yet, the clean accuracy of the proposed model using FLC pooling is about 8% better than the one reached by [43], with a 1% improvement in robust accuracy. For example, [11] has an increased training time by factor four compared to our model on CIFAR10 (see Table 6). This model achieves overall comparable results to ours. The model by Salman et al. [37] is trained with the training schedule from Madry et al. [31] and uses a multi-step adversarial attack for training. Since there is no release of the training script of this model on ImageNet, we can only roughly estimate their training times. Since they adopt the training schedule from Madry et al., we assume a similar training time increase by a factor of four, which is similar to the multi-step times reported for PGD in Table 6.

#### 4.4 Model Confidences

In Table 6, we also evaluate the confidence of model predictions. We compare each model’s confidence on correctly classified clean examples to its respective

Table 5: Comparison clean and robust accuracy against AutoAttack [7] on ImageNet using ResNet-50. We compare against results from RobustBench [6].

Method	Clean	PGD $L_{\text{inf}}$ $\epsilon = \frac{4}{255}$	Training
Baseline [6]	76.52	0.00	standard
Wong et al., 2020 [43]	55.62	26.24	fast FGSM
FGSM & FLC Pooling (ours)	63.52	27.29	fast FGSM
Robustness lib, 2019 [11]	62.56	29.22	multi-step adv. training
Salman et al., 2020 [37]	64.02	34.96	multi-step adv. training

confidence on wrongly classified adversarial examples. Ideally, the confidence on the adversarial examples should be lower. The results for the different methods show that FLC yields comparably high confidence on correctly classified clean examples with a 20% gap in confidence to wrongly classified adversarial examples. In contrast, the baseline model is highly confident in both cases. Other, even state-of-the-art robustness models have on average lower confidences but are even less confident in their correct predictions on clean examples than on erroneously classified adversarial examples (e.g. MART [42] and PGD [26]). Only the model from [44] has a trade-off preferable over the one from the proposed, FLC model.

#### 4.5 Training Efficiency

Most adversarial training approaches use adversarial image perturbations during training [14,26,43]. Thereby the time and memory needed depends highly on the specific attack used to generate the perturbations. Multi-step attacks like PGD [26] require significantly more time than single step attacks like FGSM [43]. TRADES [46] incorporates different loss functions to account for a good trade-off between clean and robust accuracy. With our FLC pooling, we provide a simple and fast method for more robust models. Therefore we compare our method with state-of-the-art training schedules in terms of time needed per epoch when trained with their provided default settings.<sup>5</sup>

Table 6 shows that FGSM training is fastest. However, FGSM with early stopping is not able to maintain high robustness against AutoAttack [7] due to catastrophic overfitting. PGD training can establish robustness against AutoAttack. It relies on the same training procedure as FGSM but uses expensive multi-step perturbations and thereby increases the computation time by a factor of four. For TRADES the training time per epoch is over six times for AWP and over eight times higher. Our FLC pooling adds only seven seconds per epoch while achieving a good clean and robust accuracy.

<sup>5</sup> When we re-run [43] from their provided code, the model has different properties than their deployed model. However, this should not affect training times. We evaluate both models in subsequent tables whenever deployed models are available.

Table 6: Runtime of adversarial training in seconds per epoch over 200 epochs and a batch size of 512 trained with a PreAct-ResNet-18 for training on the original CIFAR-10 dataset without additional data. Experiments are executed on one Nvidia Tesla V100. Evaluation for clean and robust accuracy, higher is better, on AutoAttack [7] with our trained models. The models reported by the original authors may have different numbers due to different hyperparameter selection. We also report each models confidence on their correct predictions on the clean data (Clean Confidence) and the models confidence on its false predictions due to adversarial perturbations (Perturbation Confidence). The top row reports the baseline without adversarial training.

Method	Seconds per epoch (avg)	Clean Acc	AA Acc	Clean Confidence	Perturbation Confidence
Baseline	14.6 $\pm$ 0.1	95.08	0.00	100.00	97.89
FGSM & early stopping [43]	<b>27.3 <math>\pm</math> 0.1</b>	82.88	11.82	90.50	84.26
FGSM & FLC Pooling (Ours)	34.5 $\pm$ 0.1	<b>84.81</b>	38.41	<b>98.84</b>	70.98
PGD [26]	115.4 $\pm$ 0.2	83.11	40.35	56.58	75.00
Robustness lib [11]	117 $\pm$ 19.0	76.37	32.10	95.22	78.91
TRADES [46]	219.4 $\pm$ 0.5	81.49	46.91	53.94	50.46
MART [42]	180.4 $\pm$ 0.8	55.49	8.63	24.44	50.17
AWP [44]	179.4 $\pm$ 0.4	82.61	<b>49.43</b>	88.83	<b>37.98</b>

When adding additional data like the *ddpm* dataset to the training as it is done in all leading RobustBench [6] models, the training time is increased by a factor of twenty. The *ddpm* dataset incorporates one million extra samples, which is over sixteen times more than the original CIFAR-10 dataset.

## 4.6 Black Box Attacks

PGD and AutoAttack are intrinsically related to FGSM. Therefore, to allow for a clean evaluation of the model robustness without bias towards the training scheme, we also evaluate black box attacks. Squares [1], which is also part of the AutoAttack pipeline, adds perturbations in the form of squares onto the image until the label flips. Besides Squares, we evaluate two transferred perturbations. The first perturbation set is constructed using AutoAttack on the baseline network which is not robust at all. The second set is constructed from the baseline network which we trained with FGSM and early stopping, according to [43]. We evaluate against different PreAct-ResNet-18 and WRN-28-10 models on CIFAR-10 as well as PreAct-ResNet-18 models on CIFAR-100 provided by RobustBench [6]. Note that all networks marked with \* are models which rely on additional data sources such as *ddmp*[20]. Other RobustBench models like [16] rely on training data that is not available anymore such that fair comparison is currently not possible. Arguably, we always expect models to further improve as training data is added.

Table 7: Robustness against black box attacks on PreAct-ResNet-18 and WRN-28-10 models with CIFAR-10. First against Squares [1] with  $\epsilon = 1/255$  and then against perturbations which were created on the baseline network, meaning transferred perturbations (TP), and the baseline model including early stopping (TPE). As well as the accuracy under common corruptions (CC).

Model	Clean	Squares	TP	TPE	CC
<b>Preact-ResNet-18</b>					
Baseline	<b>90.81</b>	78.04	0.00	69.33	71.81
FGSM & early stopping [43]	82.88	77.58	77.67	3.76	71.80
Wong et al., 2020 [43]	83.34	80.25	82.03	78.81	74.60
Andriushchenko & Flammarion, 2020 [2]	79.84	76.78	78.65	75.06	72.05
FGSM & FLC Pooling (ours)	84.81	<b>81.40</b>	<b>83.64</b>	<b>80.49</b>	<b>76.15</b>
Rebuffi et al., 2021 [34] *	83.53	81.24	82.36	80.28	75.79
<b>WRN-28-10</b>					
Baseline	86.67	76.17	0.09	67.3	77.33
FGSM & early stopping [43]	82.29	78.01	80.8	28.54	72.55
FGSM & FLC Pooling (ours)	84.93	81.06	83.85	72.56	75.44
Carmon et al., 2019 [4] *	<b>89.69</b>	<b>87.70</b>	<b>89.12</b>	<b>83.55</b>	81.30
Hendrycks et al., 2019 [19]	87.11	85.02	86.47	80.12	85.02
Wang et al., 2020 [42] *	87.50	85.30	86.74	80.65	<b>85.30</b>
Zhang et al., 2021 [47]	89.36	87.45	88.70	83.08	80.11

Table 7 shows that for PreAct-ResNet-18 models our FLC pooling is consistently able to prevent black box attacks better as well as to maintain clean accuracy than other robust models from RobustBench. For WRN-28-10 models, we see a clear trend that models trained with additional data can achieve higher robustness. This is expected as wider networks can leverage additional data more effectively. One should note that all of these methods require different training schedules which are at least five times slower than ours and additional data which further increases the training time. For example, incorporating the *ddpm* dataset into the training increases the amount of training time by a factor of twenty. For CIFAR-100 (Table 8) our model is on par with [35].

#### 4.7 Corruption Robustness

To demonstrate that our model generalizes the concept of robustness beyond adversarial examples, we also evaluate it on common corruptions incorporated with CIFAR-C [18]. We compare our model against our baseline as well as other RobustBench [6] models. Similar to the experiments on black box adversarial attacks we distinguish between models using only CIFAR-10 training data and models using extra-data like *ddpm* (marked by \* in Table 7) shows that our FLC

Table 8: Robustness against black box attacks for PreAct-ResNet-18 on CIFAR-100. First against Squares [1] with  $\epsilon = 1/255$  and then against perturbations which were created on the baseline network, meaning transferred perturbations (TP), and the baseline model including early stopping (TPE). As well as the accuracy under common corruptions (CC).

Model	Clean	Squares	TP	TPE	CC
Baseline	51.92	45.74	11.13	23.91	41.22
FGSM & early stopping [43]	52.09	45.75	23.90	10.88	41.15
Rice et al., 2020 [35]	53.83	<b>48.92</b>	<b>45.97</b>	<b>46.11</b>	43.48
FGSM & FLC Pooling (ours)	<b>54.66</b>	48.85	45.59	45.31	<b>44.18</b>

pooling, when trained only on CIFAR-10, can outperform other adversarially robust models as well as the baseline in terms of robustness against common corruption for the PreAct-ResNet-18 architecture. As discussed above, WRN-28-10 models are designed to efficiently leverage additional data. As our model is exclusively trained on the clean CIFAR-10 dataset we can not establish the same robustness as other methods on wide networks. However, we can also see a significant boost in robustness. Table 8 reports the results for CIFAR-100. There we can see that FLC pooling not only boosts the clean accuracy but also the robust accuracy on common corruptions.

#### 4.8 Shift-Invariance

Initially, anti-aliasing in CNNs has also been discussed in the context of shift-invariance [48]. Therefore, after evaluating our model against adversarial and common corruptions, we also analyze its behavior under images shift. We compare our model with the baseline as well as the shift-invariant models [48,49].

FLC pooling can outperform all these specifically designed approaches in terms of consistency under shift, while BlurPooling [48] does not outperform the baseline. We assume that BlurPooling is optimized for larger images sizes like ImageNet, 224 by 224 pixels, compared to 32 by 32 pixels for CIFAR-10. The adaptive model from [49] is slightly better than the baseline but can not reach the consistency of our model.

## 5 Discussion & Conclusions

The problem of aliasing in CNNs or GANs has recently been widely discussed [10,22,23]. We contribute to this field by developing a fully aliasing free down-sampling layer that can be plugged into any down-sampling operation. Previous attempts in this direction are based on blurring before down-sampling. This can help to reduce aliasing but can not eliminate it. With FLC pooling we developed

Table 9: Consistency of PreAct-ResNet-18 model prediction under image shifts on CIFAR-10. Each model is trained without adversarial training with the same training schedule (see appendix for details).

Model	Clean	Consistency under shift
Baseline	94.78	86.48
BlurPooling [48]	<b>95.04</b>	86.19
adaptive BlurPooling [49]	94.97	91.47
FLC Pooling (ours)	94.66	<b>94.46</b>

a hyperparameter-free and easy plug-and-play down-sampling which supports CNNs native robustness. Thereby, we can overcome the issue of catastrophic overfitting in single-step adversarial training and provide a path to reliable and fast adversarial robustness. While our FLC pooling based model can still be fooled when the perturbation is sufficiently strong, we can show that our model predictions are significantly less confident in this case. We hope that FLC pooling will be used to evolve to fundamentally improved CNNs which do not need to account for aliasing effects anymore.

## References

1. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: European Conference on Computer Vision. pp. 484–501. Springer (2020)
2. Andriushchenko, M., Flammarion, N.: Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems* **33**, 16048–16059 (2020)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE (2017)
4. Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J.C., Liang, P.S.: Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems* **32** (2019)
5. Chen, T., Zhang, Z., Liu, S., Chang, S., Wang, Z.: Robust overfitting may be mitigated by properly learned smoothening. In: International Conference on Learning Representations (2021), <https://openreview.net/forum?id=qZzy5urZw9>
6. Croce, F., Andriushchenko, M., Sehwan, V., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. arXiv preprint arXiv:2010.09670 (2020)
7. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: ICML (2020)
8. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: Cifar-10 is not imagenet or cifar-10. arXiv preprint arXiv:1810.03505 (2018)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). <https://doi.org/10.1109/CVPR.2009.5206848>



10. Durall, R., Keuper, M., Keuper, J.: Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions (2020)
11. Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., Tsipras, D.: Robustness (python library) (2019), <https://github.com/MadryLab/robustness>
12. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)
13. Gonzalez, R.C., Woods, R.E.: Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA (2006)
14. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015)
15. Goyal, S., Qin, C., Uesato, J., Mann, T., Kohli, P.: Uncovering the limits of adversarial training against norm-bounded adversarial examples (2021)
16. Goyal, S., Rebuffi, S.A., Wiles, O., Stimberg, F., Calian, D.A., Mann, T.A.: Improving robustness using generated data. *Advances in Neural Information Processing Systems* **34** (2021)
17. Grabinski, J., Keuper, J., Keuper, M.: Aliasing coincides with CNNs vulnerability towards adversarial attacks. In: The AAAI-22 Workshop on Adversarial Machine Learning and Beyond (2022), <https://openreview.net/forum?id=vKc1mLxBebP>
18. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations* (2019)
19. Hendrycks, D., Lee, K., Mazeika, M.: Using pre-training can improve model robustness and uncertainty. In: *International Conference on Machine Learning*. pp. 2712–2721. PMLR (2019)
20. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* **33**, 6840–6851 (2020)
21. Hossain, M.T., Teng, S.W., Soheli, F., Lu, G.: Anti-aliasing deep image classifiers using novel depth adaptive blurring and activation function (2021)
22. Jung, S., Keuper, M.: Spectral distribution aware image generation. In: AAAI (2021)
23. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems* **34** (2021)
24. Kim, H., Lee, W., Lee, J.: Understanding catastrophic overfitting in single-step adversarial training (2020)
25. Krizhevsky, A.: Learning multiple layers of features from tiny images. University of Toronto (05 2012)
26. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale (2017)
27. Li, Q., Shen, L., Guo, S., Lai, Z.: Wavelet integrated cnns for noise-robust image classification. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 7245–7254 (2020)
28. Li, Q., Shen, L., Guo, S., Lai, Z.: Wavencnet: Wavelet integrated cnns to suppress aliasing effect for noise-robust image classification. *IEEE Transactions on Image Processing* **30**, 7074–7089 (2021). <https://doi.org/10.1109/tip.2021.3101395>, <http://dx.doi.org/10.1109/TIP.2021.3101395>
29. Lohn, A.J.: Downscaling attack and defense: Turning what you see back into what you get (2020)

30. Lorenz, P., Strassel, D., Keuper, M., Keuper, J.: Is robustbench/autoattack a suitable benchmark for adversarial robustness? In: The AAAI-22 Workshop on Adversarial Machine Learning and Beyond (2022), <https://openreview.net/forum?id=aLB3FaqoMBs>
31. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
32. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
33. Rade, R., Moosavi-Dezfooli, S.M.: Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In: ICML 2021 Workshop on Adversarial Machine Learning (2021), <https://openreview.net/forum?id=BuD2LmNaU3a>
34. Rebuffi, S.A., Gowal, S., Calian, D.A., Stimberg, F., Wiles, O., Mann, T.: Fixing data augmentation to improve adversarial robustness (2021)
35. Rice, L., Wong, E., Kolter, Z.: Overfitting in adversarially robust deep learning. In: International Conference on Machine Learning. pp. 8093–8104. PMLR (2020)
36. Rony, J., Hafemann, L.G., Oliveira, L.S., Ayed, I.B., Sabourin, R., Granger, E.: Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4322–4330 (2019)
37. Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., Madry, A.: Do adversarially robust imagenet models transfer better? Advances in Neural Information Processing Systems **33**, 3533–3545 (2020)
38. Schwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., Mittal, P.: Improving adversarial robustness using proxy distributions (2021)
39. Stutz, D., Hein, M., Schiele, B.: Relating adversarially robust generalization to flat minima (2021)
40. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014), <http://arxiv.org/abs/1312.6199>
41. Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., Gu, Q.: Improving adversarial robustness requires revisiting misclassified examples. In: International Conference on Learning Representations (2020), <https://openreview.net/forum?id=rk10g6EFwS>
42. Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., Gu, Q.: Improving adversarial robustness requires revisiting misclassified examples. In: International Conference on Learning Representations (2020), <https://openreview.net/forum?id=rk10g6EFwS>
43. Wong, E., Rice, L., Kolter, J.Z.: Fast is better than free: Revisiting adversarial training. In: International Conference on Learning Representations (2020), <https://openreview.net/forum?id=BJx040EFvH>
44. Wu, D., Xia, S.T., Wang, Y.: Adversarial weight perturbation helps robust generalization. Advances in Neural Information Processing Systems **33**, 2958–2969 (2020)
45. Xiao, Q., Li, K., Zhang, D., Jin, Y.: Wolf in sheep’s clothing - the downscaling attack against deep learning applications (2017)
46. Zhang, H., Yu, Y., Jiao, J., King, E.P., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: International Conference on Machine Learning (2019)

47. Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., Kankanhalli, M.: Geometry-aware instance-reweighted adversarial training. In: International Conference on Learning Representations (2021), <https://openreview.net/forum?id=iAX016Cz8ub>
48. Zhang, R.: Making convolutional networks shift-invariant again. In: ICML (2019)
49. Zou, X., Xiao, F., Yu, Z., Lee, Y.J.: Delving deeper into anti-aliasing in convnets. In: BMVC (2020)

## A Appendix

### A.1 Training Schedules

**CIFAR-10 adversarial training schedule:** For our baseline experiments on CIFAR-10, we used the PreAct-ResNet-18 as well as the WideResNet-28-10 (WRN-28-10) architecture as they give a good trade-off between complexity and feasibility. For the PreAct-ResNet-18 models, we trained for 300 epochs with a batch size of 512 and a circling learning rate schedule with the maximal learning rate 0.2 and minimal learning rate 0. We set the momentum to 0.9 and weight decay to  $5e^{-4}$ . The loss is calculated via Cross Entropy Loss and as an optimizer, we use Stochastic Gradient Descent (SGD). For the adversarial training, we used the FGSM attack with an  $\epsilon$  of 8/255 and an  $\alpha$  of 10/255 (in Fast FGSM the attack is computed for step size  $\alpha$  once and then projected to  $\epsilon$ ). For the WRN-28-10 we used a similar training schedule as for the PreAct-ResNet-18 models but used only 200 epochs and a smaller maximal learning rate of 0.08.

**CINIC-10 adversarial training schedule:** For our baseline experiments on CINIC-10 we used the PreAct-ResNet-18 architecture. We used 300 epochs, a batch size of 512 for each training run and a circling learning rate schedule with the maximal learning rate at 0.1 and minimal at 0. We set the momentum to 0.9 and weight decay to  $5e^{-4}$ . The loss is calculated via Cross Entropy Loss and as an optimizer, we use Stochastic Gradient Descent (SGD). For the adversarial training, we used the FGSM attack with an epsilon of 8/255 and an alpha of 10/255.

**CIFAR-100 adversarial training schedule:** For our baseline experiments on CIFAR-100 we used the PreAct-ResNet-18 architecture as it gives a good trade-off between complexity and feasibility. We used 300 epochs, a batch size of 512 for each training run and a circling learning rate schedule with the maximal learning rate at 0.01 and minimal at 0. We set the momentum to 0.9 and a weight decay to  $5e^{-4}$ . The loss is calculated via Cross Entropy Loss and as an optimizer, we use Stochastic Gradient Descent (SGD). For the adversarial training, we used the FGSM attack with an epsilon of 8/255 and an alpha of 10/255.

**CIFAR-10 clean training schedule:** Each model is trained without adversarial training. We used 300 epochs, a batch size of 512 for each training run and a circling learning rate schedule with the maximal learning rate at 0.2 and minimal at 0. We set the momentum to 0.9 and a weight decay to  $5e^{-4}$ . The loss is calculated via Cross Entropy Loss and as an optimizer, we use Stochastic Gradient Descent (SGD).

**ImageNet adversarial training schedule:** For our experiment on ImageNet (see Appendix 4.3) we used the ResNet50 architecture. We trained for 150 epochs with a batch size of 400, and a multistep learning rate schedule with an initial learning rate 0.1,  $\gamma = 0.1$ , and milestones [30, 60, 90, 120]. We set the momentum to 0.9 and weight decay to  $5e^{-4}$ . The loss is calculated via Cross Entropy Loss and as an optimizer, we use Stochastic Gradient Descent (SGD). For the adversarial training, we used FGSM attack with an epsilon of 4/255 and an alpha of 5/255.

## A.2 Aliasing Free Down-Sampling

Previous approaches like [48,49] have proposed to apply blurring operations before downsampling, with the purpose of achieving models with improved shift invariance. Therefore, they apply Gaussian blurring directly on the feature maps via convolution. In the following, we briefly discuss why this setting can not guarantee to prevent aliasing in the feature maps, even if large convolutional kernels would be applied, and why, in contrast, the proposed FLC pooling can guarantee to prevent aliasing.

To prevent aliasing, the feature maps need to be band-limited before down-sampling [13]. This band limitation is needed to ensure that after down-sampling no replica of the frequency spectrum overlap (see Figure 2). To guarantee the required band limitation for sub-sampling with a factor of two to  $N/2$  where  $N$  is the size of the original signal, one has to remove (reduce to zero) all frequency components above  $N/2$ .

**Spatial Filtering based Approaches** [48,49] propose to apply approximated Gaussian filter kernels to the feature map. This operation is motivated by the fact that an actual Gaussian in the spatial domain corresponds to a Gaussian in the frequency (e.g. Fourier) domain. As the standard deviation of the Gaussian in the spatial domain increases, the standard deviation of its frequency representation decreases. Yet, the Gaussian distribution has infinite support, regardless of its standard deviation, i.e. the function never actually drops to zero. The convolution in the spatial domain corresponds to the point-wise multiplication in the frequency domain.

Therefore, even after convolving a signal with a perfect Gaussian filter with large standard deviation (and infinite support), all frequency components that were  $\neq 0$  before the convolution will be afterwards (although smaller in magnitude). Specifically, the convolution with a Gaussian (even in theoretically ideal settings), can reduce the apparent aliasing but some amount of aliasing will always persist. In practice, these ideal settings are not given: Prior works such as [48,49] have to employ approximated Gaussian filters with finite support (usually not larger than  $7 \times 7$ ).

**FLC Pooling** Therefore, FLC pooling operates directly in the frequency domain, where it removes all frequencies that can cause aliases.

This operation in the Fourier domain is called the *ideal low pass filter* and corresponds to a point-wise multiplication of the Fourier transform of the feature maps with a rectangular pulse  $H(\hat{m}, \hat{n})$ .

$$H(\hat{m}, \hat{n}) = \begin{cases} 1 & \text{for all } \hat{m}, \hat{n} \text{ below } M/2 \text{ and } N/2 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

This trivially guarantees all frequencies above below  $M/2$  and  $N/2$  to be zero.

Table 10: Model confidences for robust models PreAct-ResNet-18 for CIFAR-10 from RobustBench compared to the baseline and our FLC Pooling. [6].

Model	Clean Acc	AA Acc	Clean Confidence	Perturbation Confidence
Baseline	95.08	0.00	100.00	97.89
FGSM & early stopping [43]	82.88	11.82	90.50	84.26
FGSM & FLC Pooling (Ours)	<b>84.81</b>	38.41	<b>98.84</b>	70.98
Andriushchenko and Flammarion, 2020 [2]	79.84	43.94	71.93	41.27
Wong et al., 2020 [43]	83.34	43.20	75.30	44.76
Rebuffi et al., 2021 [34] *	83.53	<b>56.65</b>	64.64	<b>37.07</b>

**Could we apply FLC Pooling as Convolution in the Spatial Domain?** In the spatial domain, the ideal low pass filter operation from above corresponds to a convolution of the feature maps with the Fourier transform of the rectangular pulse  $H(\hat{m}, \hat{n})$  (by the Convolution Theorem, e.g.[13]). The Fourier transform of the rectangle function is

$$\text{sinc}(m, n) = \begin{cases} \frac{\sin(\sqrt{m^2+n^2})}{\sqrt{m^2+n^2}} & m, n \neq 0 \\ 1 & m, n = 0 \end{cases} \quad (8)$$

However, while the ideal low pass filter in the Fourier domain has finite support, specifically all frequencies above  $N/2$  are zero,  $\text{sinc}(m, n)$  in the spatial domain has infinite support. Hence, we need an infinitely large convolution kernel to apply perfect low pass filtering in the spatial domain. This is obviously not possible in practice. In CNNs the standard kernel size is 3x3 and one hardly applies kernels larger than 7x7 in CNNs.

### A.3 Confidences of RobustBench Models

In Table 6 in the main paper, we also report the confidence in the decisions of our model, for a PreAct-ResNet-18 trained on CIFAR-10. Thereby, we report the average confidence of models for their correct decisions on clean data as well as their confidence for erroneous decisions on adversarial attacks. Ideally, models should be highly confident in the first case and significantly less confident in the second case. In the following, we report such evaluations for our models on CIFAR-100 and include robust models from other state-of-the-art methods trained by the original authors and made available on RobustBench [6].

The overall trend which can be seen in Table 10, 11, and 12 demonstrates that robust models are much less confident in their false predictions due to adversarial attacks than non-robust models. Indicating that robust models not only exhibit robustness against adversarial attacks but also strong indication through the confidence that a false prediction is indeed wrong.

Table 11: Model confidences for robust models WRN-28-10 from RobustBench compared to the baseline and our FLC Pooling. [6].

Model	Clean Acc	AA Acc	Clean Confidence	Perturbation Confidence
Baseline	86.67	0.09	95.29	95.86
FGSM & early stopping	82.29	28.54	84.95	66.69
FGSM & FLC Pooling (Ours)	84.93	38.37	<b>95.86</b>	87.81
Hendrycks et al., 2019, [19]	87.11	54.89	75.71	42.32
Wang et al., 2020 [42]	87.50	56.32	69.02	<b>31.12</b>
Zhang et al., 2021 [47]	<b>89.36</b>	<b>59.23</b>	75.75	34.56

Table 12: Model confidences for robust models PreAct-ResNet-18 for CIFAR-100 from RobustBench compared to the baseline and our FLC Pooling. [6].

Model	Clean Acc	AA Acc	Clean Confidence	Perturbation Confidence
Baseline	51.92	11.13	82.68	65.65
FGSM & early stopping	52.09	10.88	82.27	65.42
FGSM & FLC Pooling (Ours)	<b>54.66</b>	15.40	<b>82.44</b>	60.74
Rice et al., 2020 [35]	53.83	<b>18.99</b>	73.04	<b>43.12</b>

Furthermore, we can observe that all robust models reported on RobustBench are less confident in their clean prediction than the baseline or our FLC pooling. All these models are optimized to achieve a gain in robustness by adding a significant amount of adversarial or additional data and therefore are less certain in their overall predictions. In contrast, our FLC pooling can achieve high confidence on its clean accuracy while achieving robustness and lower confidence on its false prediction.

Figure 5 demonstrates the confidence distribution of the different models even clearer. We can see that robust models only trained with specific training schedules can be equally confident in their true and false predictions. Our FLC pooling, in contrast, is able to correctly classify samples with high confidence and it is less confident on its wrong predictions.

In Figure 6, we evaluate how well calibrated the confidences of our model are, in comparison to state-of-the-art robust models. Therefore, we classify the predictions of the models by a threshold on their confidence to determine whether a prediction is correct or wrong. Higher confidences should correspond to correct predictions. The resulting precision and recall curves indicate that our proposed model is best calibrated for predictions on clean data and performs on par with other adversarially robust models on adversarial data.

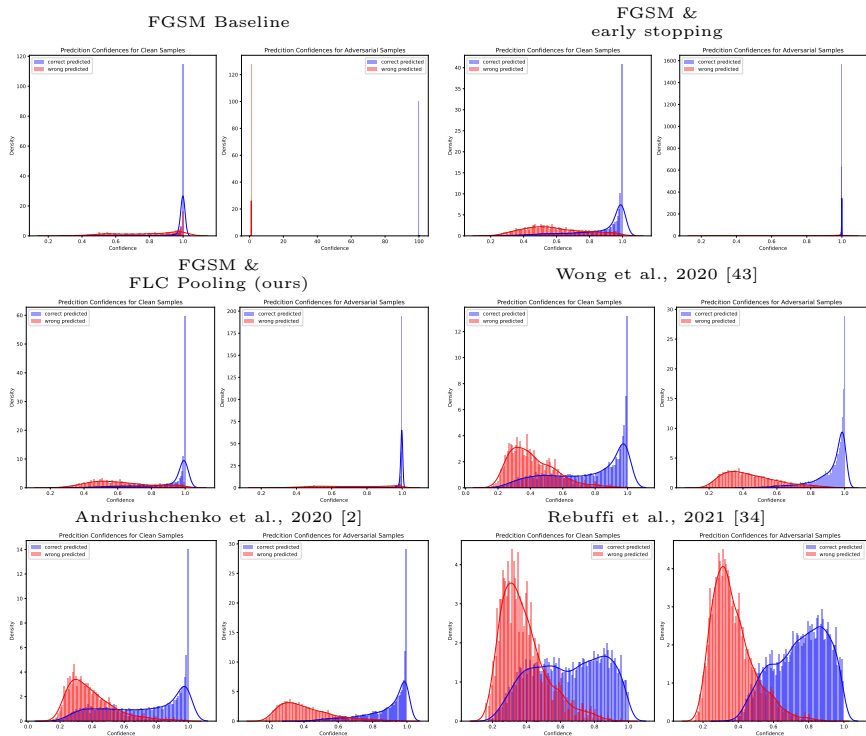


Fig. 5: Here, we plot the distribution of the confidences for each model on the clean and adversarial samples. We separate the results into clean (left) and adversarial samples (right) and in each plot into correctly (blue) and incorrectly (red) classified samples. Our model with FLC pooling exhibits strong confidence on its correct predictions on the clean data similar to the baseline with and without early stopping. Models taken from RobustBench [6] are much less confident in their clean prediction. Further, the confidences from robust models from RobustBench [6] show a wider spread in confidences, especially the model provided by Rebuffi et al. [34].



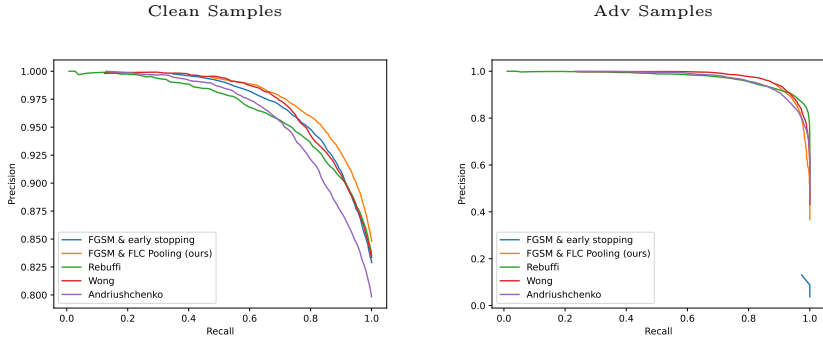


Fig. 6: Precision-Recall-Curves for classification by confidence. Left: We classify by a threshold on the model confidence between correct and wrong classifications on clean data. The proposed model has the best calibrated confidences of all robust models. Right: We classify by a threshold on the confidences between correct and wrong predictions on adversarial data. Our model performs on par with the state-of-the-art robust models while the model achieved by early stopping yields very poor performance.

#### A.4 AutoAttack Attack Structure

In the main paper we showed one example of an image optimized by AutoAttack [7] to fool our model and the baseline in Figure 4. In Figure 7, we give more examples for better visualisation and comparison.

#### A.5 Ablation Study: Additional Frequency Components

In addition to the low frequency components we tested different settings in which we establish a second path through which we aim to add high frequency or the original information. We either add up the feature maps or contacted them. The procedure of how to include a second path is represented in Figure 8. One approach is to execute the standard down-sampling and add it to the FLC pooled feature map. The other is to perform a high pass filter on the feature map and down-sample these feature maps. Afterwards, the FLC pooled feature maps as well as the high pass filtered and down-sampled ones are added. With this ablation, we aim to see if we do lose too much through the aggressive FLC pooling and if we would need additional high frequency information which is discarded through the FLC pooling. Table 13 show that we can gain minor points for the clean but not for the robust accuracy. Hence we did not see any improvement in the robustness and an increase in training time per epoch as well as a minor increase in model size, we will stick to the simple FLC pooling.

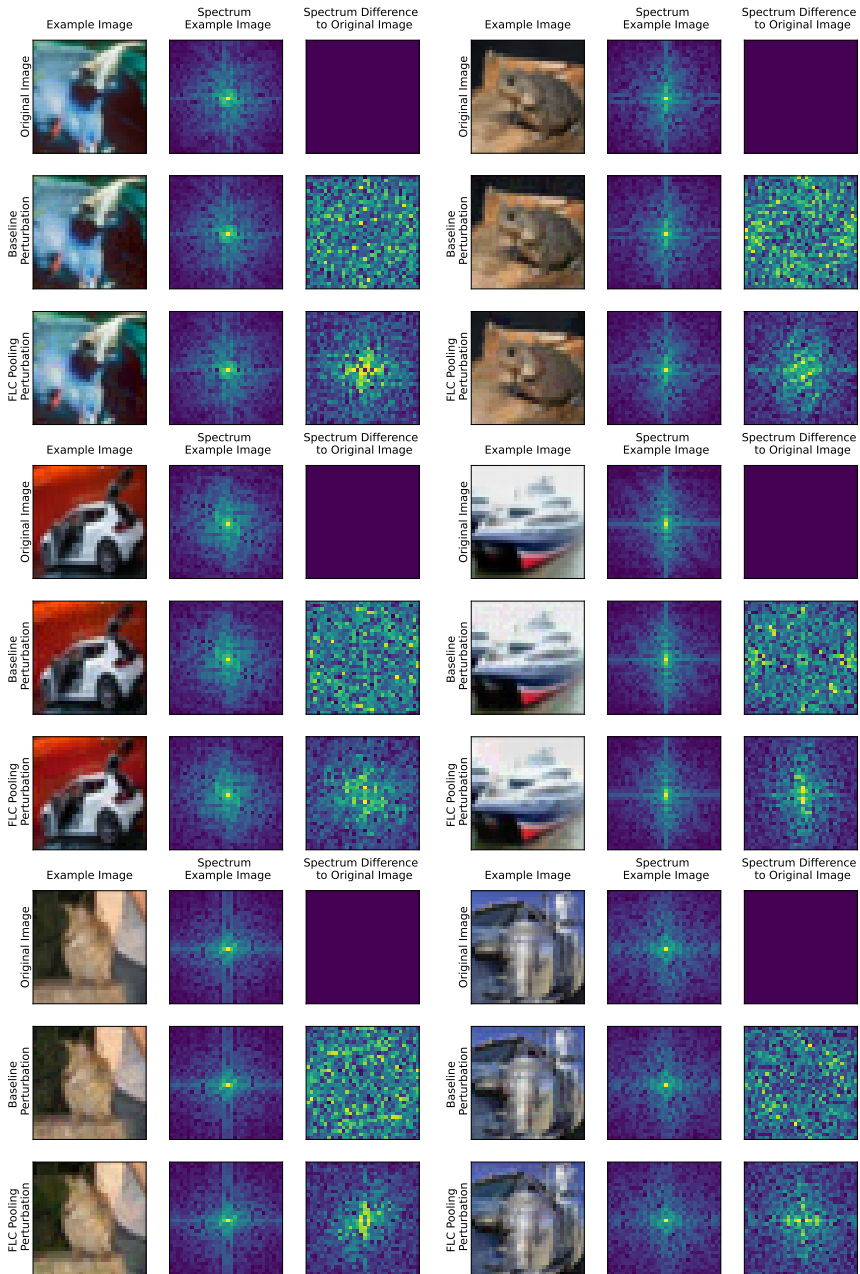


Fig. 7: Spectrum and spectral differences of adversarial perturbations created by AutoAttack with  $\epsilon = \frac{8}{255}$  on the baseline model as well as our FLC Pooling. The classes from top left down to the bottom right are: Bird, Frog, Automobile, Ship, Cat and Truck.

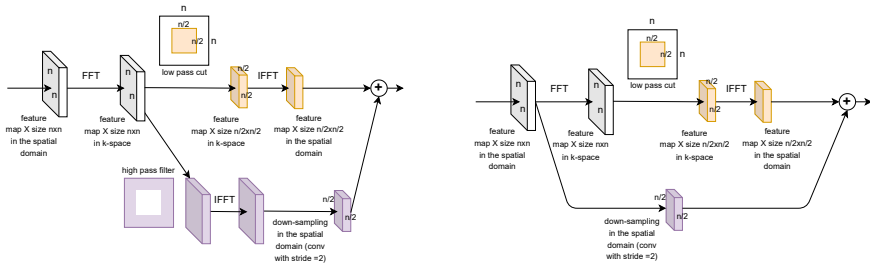


Fig. 8: FLC pooling plus, which either includes the original down-sampled signal like it is done traditionally (right) or with the high frequency components filtered by a high pass filter in the Fourier domain and down-sampled in the spatial domain by an identity convolution of stride two (left).

Table 13: Accuracies for CIFAR-10 Baseline LowCutPooling plus the original or high frequency part of the featuremaps down-sampled in the spatial domain for FGSM Training. We can see that the additional data does not improve the robust accuracy and gives only minor improvement for the clean accuracy. Due to the additional computations necessary for the high frequency /original part we decided to fully discard them and stick to the pure low frequency cutting.

Method	Clean	PGD $L_{inf}$ $\epsilon = \frac{8}{255}$	Seconds per epoch (avg)	Model size (MB)
FLC pooling	84.81	38.41	$34.6 \pm 0.1$	42.648
FLC pooling + HighPass pooling	85.38	38.02	$45.2 \pm 0.4$	42.652
FLC pooling + Original pooling	85.37	38.30	$35.4 \pm 0.1$	42.652