INHIBITION-AUGMENTED CONVNETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Networks (ConvNets) suffer from insufficient robustness to common corruptions and perturbations of the input, unseen during training. We address this problem by including a form of response inhibition in the processing of the early layers of existing ConvNets and assess the resulting representation power and generalization on corrupted inputs. The considered inhibition mechanism consists of a non-linear computation that is inspired by the push-pull inhibition exhibited by some neurons in the visual system of the brain. In practice, each convolutional filter (push) in the early layers of conventional ConvNets is coupled with another filter (pull), which responds to the preferred pattern of the corresponding push filter but of opposite contrast. The rectified responses of the push and pull filter pairs are then combined by a linear function. This results in a representation that suppresses responses to noisy patterns (e.g. texture, Gaussian, shot, distortion, and others) and accentuates responses to preferred patterns.

We deploy the layer into existing architectures, (Wide-)ResNet and DenseNet, and propose new residual and dense push-pull layers. We demonstrate that ConvNets that embed this inhibition into the initial layers are able to learn representations that are robust to several types of input corruptions. We validated the approach on the ImageNet and CIFAR data sets and their corrupted and perturbed versions, ImageNet-C/P and CIFAR-C/P. It turns out that the push-pull inhibition enhances the overall robustness and generalization of ConvNets to corrupted and perturbed input data. Besides the improvement in generalization, notable is the fact that ConvNets with push-pull inhibition have a sparser representation than conventional ones without inhibition. The code and trained models will be made available.

1 INTRODUCTION

After AlexNet was proposed, several other more sophisticated Convolutional Networks (ConvNets), e.g. VGGNet (Simonyan & Zisserman, 2014), GoogleNet (Szegedy et al., 2015), ResNet (He et al., 2015), DenseNet (Huang et al., 2017), consecutively achieved lower error rates on classification benchmarks such as CIFAR (Krizhevsky, 2009), SVHN (Netzer et al., 2011) and ImageNet (Krizhevsky et al., 2012). The witnessed reduction of the classification error, however, was not accompanied by a similar increase of generalization and robustness to common corruptions or perturbations in the test images. When test images contain artefacts that are not present in the training data (e.g. shot noise, JPEG compression, blur), the performance of SOTA networks, like ResNet or DenseNet, degrade relatively more than that of AlexNet (Hendrycks & Dietterich, 2019).

Robustness of DeepNets and ConvNets is gaining attention from researchers in machine learning and computer vision. One point of analysis concerns adversarial attacks that consist of very small, visually imperceptible perturbations of the input signals such that confuse the classification model (Akhtar & Mian, 2018). Several adversarial attacks were proposed based on the knowledge of the classification models (Goodfellow et al., 2015) or on iterative methods (Kurakin et al., 2016; Madry et al., 2018). Class-specific and universal black box attacks (Moosavi-Dezfooli et al., 2017) were also proposed. As new types of adversarial perturbations are designed, defense algorithms need to be developed (Lu et al., 2017; Metzen et al., 2017).

In contrast to adversarial perturbations, common corruptions and perturbations are modifications of the input signals due to typical artefacts generated by sensor noise, illumination changes, transformations determined by changes of camera perspective (e.g. rotations or elastic transformations), quality loss due to compression, among others. These can be considered as average cases of ad-

versarial attacks and are very common in computer vision tasks. In this study, we focus on the robustness of ConvNets to common corruptions and perturbations and demonstrate how the use of inhibition-augmented filters increases the robustness of existing ConvNet models. We use a layer that implements a local-response suppression mechanism (Strisciuglio et al., 2020), named push-pull, to replace some convolutional layers in existing ConvNets. The design of this layer was inspired by neurophysiological evidence of the push-pull inhibition exhibited by some neurons in the early part of the visual system of the brain (Hirsch et al., 1998). Lauritzen & Miller (2003) stated that *'push-pull inhibition [...] acts to sharpen spatial frequency tuning, [...] and increase the stability of cortical activity'*. These neurons have inhibitory interneurons with receptive fields of opposite polarity. The interneurons suppress the responses of the associated neurons in noisy local patterns; that is they pull the push responses. From an engineering point of view, the push-pull inhibition can be considered as a band-pass filter. A computational model of the push-pull inhibition was introduced in image processing operators for contour and line (Azzopardi et al., 2014; Strisciuglio et al., 2019) detection that are robust to various types of noise. Early results on augmenting ConvNets with push-pull inhibition were reported in (Strisciuglio et al., 2020), where only the first convolutional layer of existing networks was replaced and preliminary experiments on the MNIST and CIFAR images were reported.

We deploy a push-pull layer in the state-of-the-art (wide-)ResNet and DenseNet architectures by replacing the convolutional layer in the entry layer of the networks and all convolutional layers inside the first residual or dense block. The number of layers and parameters of the modified networks remains the same of their original counterpart. We train several models with and without the push-pull layers, using the images from ImageNet and CIFAR training sets, and test them on the ImageNet-C/P and CIFAR-10-C/P benchmark test sets (Hendrycks & Dietterich, 2019), which contain images with several types of corruption and perturbation. Furthermore, we combine the push-pull layer with other techniques to improve the robustness performance of ConvNets, namely the data augmentation techniques AutoAugment (Cubuk et al., 2018) and cutout (Devries & Taylor, 2017), and low-pass anti-aliasing filters (Zhang, 2019), of which we provide more details in Section 2. We show how combined data- and architecture-related actions have jointly a positive effect to further improve the robustness and generalization performance of existing models.

Our contributions are twofold: *a*) new residual and dense layers in which push-pull filters replace the original convolutions, *b*) a thorough analysis of the impact of the push-pull inhibition on the robustness of existing models, and of its combination with other strategies for model robustness.

The rest of the paper is organized as follows. We discuss related works in Section 2 and provide details of the push-pull layer and modified residual and dense layers in Section 3. We report the experiments and results in Section 4 and Section 5. Finally, we draw conclusions in Section 6.

2 Related works

One common approach to increase the generalization of existing models is data augmentation. It provides a mechanism to increase the robustness to certain transformations included in the training process. One type of data augmentation, namely added Gaussian noise, does not guarantee robustness to other types of noise, e.g. shot noise (Zheng et al., 2016). In Vasiljevic et al. (2016), for instance, it was demonstrated that using one blurring type for data augmentation during training does not result in a model that is robust to other types of blurring. Moreover, in Geirhos et al. (2018), it was observed that heavy data augmentation can cause underfitting. Devries & Taylor (2017) proposed a technique named *cutout* that masks out random square patches of the training images and increases network performance. Although cutout reduces the classification error on benchmark test sets, we noticed that it does not improve robustness to common corruptions and perturbations. Lopes et al. (2019) proposed PatchGaussian, which combines the regularization abilities of cutout with data augmentation using Gaussian noise. Randomly located patches are selected from the input images and corrupted with Gaussian noise. Autoaugment (Cubuk et al., 2018; Lim et al., 2019) is a very effective strategy to learn an optimal set of augmentation policies from training data. Together with increased classification rate on benchmark test sets, it was shown to improve the robustness of the concerned models to various corruptions and perturbations (Yin et al., 2019). Recently, AugMix was proposed. It uses diverse augmentations and a Jensen-Shannon divergence consistency loss function to train models that have better generalization and uncertainty estimates (Hendrycks et al., 2020).

Data augmentation is in some cases very powerful. However, these techniques work around architectural or structural weaknesses of ConvNets. For example, using high-frequency noise for data augmentation (e.g. Gaussian noise) makes the network focuses on low frequency features that are discriminant instead of becoming explicitly robust to high-frequency corruptions (Yin et al., 2019). In order to improve the intrinsic robustness of ConvNets, it would be preferable to deploy architectural elements that are able to better detect the features of interest also when the input data is corrupted. For instance, common downsampling methods, like max-pooling and strided convolutions, introduce aliasing as they ignore the sampling theorem of Nyquist. Zhang (2019) used a low-pass filter before down-sampling and showed that it improves classification results on ImageNet and robustness to common corruptions of the input images. A layer of Gabor filters was recently shown to increase network robustness to adversarial attacks (Pérez et al., 2020).

3 PUSH-PULL INHIBITION IN CONVNETS

A ConvNet constitutes of L layers, each of which performs a non-linear transformation $H(x)$ of the input x. In some architectures, $H(x)$ is a composite function of convolutions, batch normalization (BN), rectifier linear unit (ReLU) and pooling. For example, a residual layer contains two convolutions, batch normalizations, ReLUs and one identity connection. We use a composite layer, called push-pull, that consists of two convolutions with kernels of different size and two ReLU functions.

3.1 PUSH-PULL LAYER

The response of the push-pull layer (Figure 1) is defined as the linear combination of the rectified responses of two convolutions of the input signal, one with an excitatory (push) kernel $k(\cdot)$ and one with an inhibitory (pull) kernel $-\hat{k}(\cdot)$ (Strisciuglio et al., 2020). The responses of the push and pull components are rectified by means of the ReLU function, which introduces non-linearity, and combined: the pull response is subtracted from (i.e. it suppresses) the push response. In the forward step of the learning algorithm, the weights of each pull kernel are computed by upsampling and negating the weights of the corresponding push kernel. Since the pull kernels are directly inferred from the push kernels, the number of trainable parameters does not change. In the backward step, however, the implementation of the push-pull layer ensures that the gradient is back-propagated to both push and pull components. While only the weights of the push kernels are updated, the effect of the gradient back-propagation incorporates the pull responses. We compute the response R of a push-pull layer to an input x as:

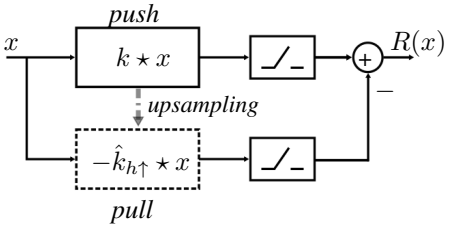

Figure 1: Structure of the push-pull layer. An input image x is convolved with a push and a pull kernel. The pull kernel is derived from the push one by negation and upsampling (by a factor h). Their responses are rectified and combined linearly.

$$R(x) = \Theta\left(x \star k\right) - \alpha\Theta\left(x \star (-\hat{k}_{h\uparrow})\right) \tag{1}$$

where $\Theta(\cdot)$ is the ReLU function, k is the push kernel and $\hat{k}_{h\uparrow}$ is its upsampled (by a factor h) and negated version, i.e. the pull kernel. We set $h = 2$, that is the pull kernel has double width and height of the push kernel. The parameter α is a weighting factor for the inhibition response, which we set to 1 (i.e. same relevance as the push response). The decision of having pull kernels larger than the push kernels is motivated by neurophysiological findings (Li et al., 2012).

The combined effect of the push and pull kernels results in an increased robustness to the detection of features of interest in corrupted input signals. Figure 2 illustrates an example of the response maps of a conventional convolution filter and those of a push-pull filter on images corrupted with noise. The response maps of the push-pull filter on corrupted images correlate better with that of the clean image in comparison to the corresponding response maps of the convolution filter. For this illustration we selected a convolutional and a push kernel that have the same shape and whose parameters are learned in the first layer of a ResNet model, without and with push-pull layers

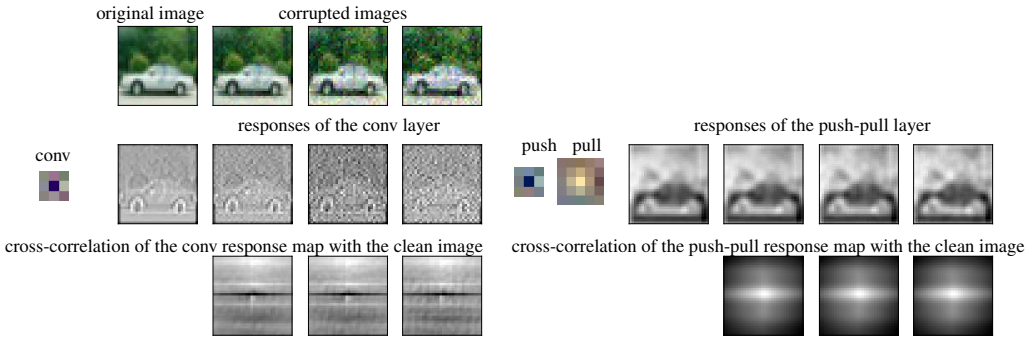

Figure 2: Illustration of robustness to noise. (Top left) An image from the CIFAR-10 test set and its corrupted versions with Gaussian noise of increasing severity. (Bottom left) The response maps of the convolutional kernel are affected by high frequency noise. (Right) The response maps of the push-pull kernel are less sensitive to the image corruption.

respectively, on the CIFAR data set. This allows to better and more directly demonstrate the effect that the pull inhibitory component has on the quality of the computed feature maps. The spatial frequency selectivity of the push-pull filter is sharper than that of a conventional convolutional layer, as it focuses on the frequency of interest and suppresses those outside a certain learned sub-band. In this way, noise induced artefacts can be filtered out and clearer feature maps are computed for further processing in subsequent layers of the network.

3.2 Push-pull in residual and dense layers

We extended the implementation of the residual and dense composite layers, originally proposed by He et al. (2015) and Huang et al. (2017), respectively. In both cases, we substituted the convolutional layers with push-pull layers, which resulted in the push-pull residual layer and the push-pull dense layer.

The push-pull inhibition is a phenomenon that is exhibited by certain neurons in the early visual system. Evidence of it has been observed in area V1 of the visual cortex (Lauritzen & Miller, 2003), which can be compared to the early layers of a ConvNet. Thus, motivated by neuro-scientific findings, we deploy the modified push-pull residual and dense layers in the first block of the considered architectures. In principle, the push-pull layer may be used to substitute a convolutional layer at any depth in a ConvNet. In practice, however, we experienced that deploying push-pull layers at deeper layers considerably changes the training dynamics of the model, not contributing to result improvement, and the nominal hyperparameters used to train the original network need to be changed.

4 Experiments

In the following, we refer as ResNet-L to a residual network with L layers (He et al., 2015) and as DenseNet-L-k to a densely connected network with L layers and growing factor k (Huang et al., 2017). The notation '-pp' in the model name indicates the use of the push-pull layer that replaces the first convolutional layer. We use the suffix '-b1' to denote that the concerned model deploys push-pull residual or dense layers in the first block of the network as well.

4.1 Data sets

We trained several models on the CIFAR and ImageNet data sets, and tested them on the original test sets and on those provided with the CIFAR-C/P and ImageNet-C/P benchmarks (Hendrycks & Dietterich, 2019). The CIFAR-C and ImageNet-C data sets are composed of 75 test sets of corrupted images, generated by applying 15 types of corruption with 5 levels of severity, to the original images. The corruptions are arranged in four categories: noise (Gaussian, shot and impulse noise), blur (defocus, glass, motion and zoom blur), weather (snow, frost, fog, brightness) and digital distortions (contrast, elastic transformation, pixelate, jpeg compression). The perturbations in CIFAR-P and ImageNet-P have a character of temporality. Starting from the corrupted images in CIFAR-C and ImageNet-C, sequences of 30 frames were generated by applying consecutive small perturbations. For each image in the original CIFAR and ImageNet validation sets, ten types of perturbed sequences

were created (Gaussian and shot noise, motion and zoom blur, snow, brightness, translate, rotate, tilt, scale). More details about the -C and -P data sets are reported by Hendrycks & Dietterich (2019).

4.2 EVALUATION

We applied the evaluation protocol proposed by Hendrycks & Dietterich (2019). Besides the classification error on the CIFAR and ImageNet, we evaluate the corruption error C, i.e. the classification error achieved on the CIFAR-C and ImageNet-C sets, and the flip-probability FP, that is the probability that the outcome of the classification changes between consecutive frames, on the sequences in the CIFAR-P and ImageNet-P sets. We compute the mean and relative corruption errors mCE and rCE and the normalized flip probability, i.e. the flip rate FR. For details about these metrics, we refer the reader to Appendix B and to the paper of Hendrycks & Dietterich (2019).

4.3 EXPERIMENTS

CIFAR. We trained several ResNet and DenseNet models and their versions with the push-pull layer, using the images from the CIFAR training set to which we applied a light data augmentation, consisting only of center-cropping and random horizontal flipping (Lee et al., 2015). For both (wide)ResNet- and DenseNet-based models we optimized the parameters with stochastic gradient descent (SGD), Nesterov momentum equals to 0.9 and weight decay equals to 10^{-4}. We trained ResNet architectures for 200 epochs, with batch size equals to 128 and initial learning rate of 0.1, which we decreased by a factor of 10 at epochs 80, 120 and 160. We trained DenseNet models for 350 epochs with batch size of 64 and initial learning rate of 0.1, which we decreased by a factor of 10 at epochs 150, 225 and 300. These hyperparameters are the same used to train the original ResNet and DenseNet models. The use of the push-pull layer in the first block of existing architectures does not require to change the hyperparameters used to train the original networks.

Furthermore, we experimented by enhancing the considered architectures with the anti-aliased sampling layers proposed by Zhang (2019) for improvement of shift-invariance. We compared the performance of what the push-pull layer and the anti-aliased sampling layer, and showed that their combination contributes to a further improvement of the robustness of ConvNets. We experimented with anti-aliasing filters of size 2×2, 3×3 and 5×5. We also trained WideResNet models (Zagoruyko & Komodakis, 2016) with and without the push-pull layer, using different data augmentations, namely cutout (Devries & Taylor, 2017), AutoAugment (Cubuk et al., 2018), and cutout plus AutoAugment. We show that the push-pull layer, although already providing the network with intrinsic robustness to corruptions, can be combined with data augmentation to achieve further robustness.

ImageNet. We considered ResNet-18 and ResNet-50 models in which we deployed the push-pull layer as a replacement of the first convolutional layer and of the convolutions inside the first residual block. In the case of ResNet-50, we replaced only the middle convolution of the bottleneck layers. We trained the networks using SGD optimization with Nesterov moment equals to 0.9 and weight decay of $5 \cdot 10^{-4}$ for 90 epochs, with an initial learning rate of 0.1 that we decreased by a factor of 10 after 30, 60 and 85 epochs. We used a batch size of 256. For comparison purposes, we trained ResNet-18 and ResNet-50 models augmented with the anti-aliased sampling layers proposed by Zhang (2019) using the same hyperparameters and schedule of the learning rate that we employed for the training of the push-pull networks. For ResNet-50 with anti-aliasing, due to GPU memory limitations, we reduced the batch size to 128.

5 RESULTS AND DISCUSSION

5.1 CIFAR

Effect of the push-pull inhibition layer. In Table 1, we report the results of ResNet and DenseNet networks on the CIFAR and CIFAR-C/P test sets. The lower the numbers the better the performance. We tested architectures with different number of layers (and growing factors, for DenseNet). For each architecture, we trained the original model with convolutional layers and two models with the push-pull layer, one (-pp) with the push-pull layer in the first layer only and the other (-b1) with the push-pull layer replacing all convolutions in the first block of the network as well. The

Table 1: Results obtained on the CIFAR-C and CIFAR-P test sets by networks that embed the push-pull layer. For each push-pull model, the results are to be compared with its baseline network without inhibition. E indicates the classification error on the original test set. The mCE, rCE and mFR are normalized by the corresponding values achieved by the baseline AlexNet.

Results on CIFAR(-C/P)

Network	E	mCE	rCE	mFR
AlexNet	13.87	1	1	1
ResNet-20	**7.56**	1.07	1.79	1.06
ResNet-20-pp	8.29	1.04	1.68	1.07
ResNet-20-b1	8.32	**0.98**	**1.52**	**0.96**
ResNet-44	**6.76**	0.99	1.60	0.94
ResNet-44-pp	6.87	0.94	1.48	0.90
ResNet-44-b1	7.18	**0.84**	**1.25**	**0.76**
ResNet-56	**6.64**	1.00	1.72	0.90
ResNet-56-pp	7.01	0.93	1.54	0.85
ResNet-56-b1	7.76	**0.86**	**1.31**	**0.82**
DenseNet-40-12	**6.33**	1.05	1.77	1.00
DenseNet-40-12-pp	7.13	1.01	1.64	**0.967**
DenseNet-40-12-b1	7.16	**0.98**	**1.59**	0.97
DenseNet-100-24	**3.88**	0.81	1.44	0.66
DenseNet-100-24-pp	4.5	0.75	1.28	0.61
DenseNet-100-24-b1	4.27	**0.73**	**1.27**	**0.57**

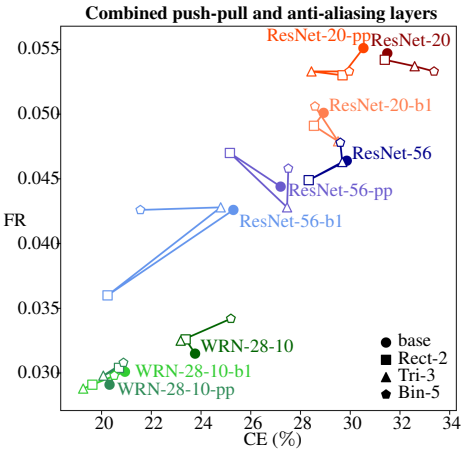

Figure 3: Results achieved by combining the push-pull layer with the anti-aliasing filters proposed by Zhang (2019). We group by colors the results achieved by a certain model with different anti-aliasing filter configurations. The closer the points are to the bottom-left the better the robustness is.

measurements mCE, rCE and mFR (see Appendix B) are normalized with respect to the results of the baseline AlexNet.

The push-pull layer contributes to a substantial improvement of the robustness of the networks to corruptions and perturbations of the input images. Its effect, especially when it is deployed in the first block of existing ConvNets, is noticeable, with a reduction of the mean corruption error mCE by more than 10% in some cases. The improvement in robustness trades off with a generally small reduction of the classification accuracy on original (clean) test data. The presence of the push-pull layer has an important effect in reducing the relative corruption error rCE, i.e. the average gap between the classification error on clean data (on which the model is trained) and corrupted data. It indicates that models with push-pull inhibition are better in generalizing to data with heavy corruptions. In some cases, the relative error is decreased by more than 30%. The flip rate (mFR), computed on the CIFAR-P sequences, also improved for all models with push-pull layers.

Networks with the push-pull layers generally show an improvement of robustness to corruptions with high-frequency components. For instance, ResNet20-pp and ResNet20-b1 reduced the CE by 10% to 25% on noise (Gaussian, shot, impulse), up to 18% on glass blur, and between 15% and 20% on pixelate and jpeg compression corruptions with respect to that of the original ResNet20. Marginal improvements or lower robustness are obtained on low-frequency corruptions, such as defocus, motion and zoom blur ($+2/6\%$ of CE), snow, frost and fog (between -4% and $+4\%$ of CE). For detailed results per corruption and perturbation type, we refer the reader to Appendix C.

Inhibition and anti-aliasing layers. In order to further improve the robustness of existing ConvNets, we show that the push-pull inhibition can be used in combination with other techniques. We couple it with the anti-aliased sampling layer proposed by Zhang (2019), which showed to improve robustness to some image corruptions next to re-enforcing the shift-invariance property of ConvNets. It consists of a low-pass filter inserted before the pooling or strided convolutional operations. We used low-pass filters of different sizes, namely 2×2 (Rect-2), 3×3 (Tri-3) and 5×5 (Bin-5).

Figure 3 illustrates the results achieved by (Wide)ResNet models extended with the push-pull inhibition and anti-aliased sampling layers. In many cases the combined effects of the push-pull and anti-aliasing layers resulted in lowering either the corruption error CE or the flip probability FP. The Rect-2 filter (□ marker), in particular, is effective on all models (with and without the push-pull layer), on the contrary of the Tri-3 and Bin-5 filters, which have larger sizes and in some cases worsen the robustness of the networks. This is due to the relatively small sizes of the images in the

Table 2: Corruption error (CE) and flip rate (FR) of WideResNet models with and without the push-pull layer, trained using different data augmentations.

		Augmentation			
	Model	**none**	**AutoAug.**	**cutout**	**AutoAug.+cutout**
mCE	WRN-28-10	0.796	0.523	0.814	0.535
	*-pp	0.689	0.521	0.804	0.548
	*-b1	0.703	0.499	0.728	**0.484**
FR	WRN-28-10	0.602	0.408	0.611	0.391
	*-pp	0.555	0.445	0.567	0.417
	*-b1	0.572	0.428	0.529	**0.384**

Table 3: Results obtained on the ImageNet-C and ImageNet-P by ConvNets that embed the push-pull layer as a substitute of different convolutional layers. For each augmented model with push-pull and anti-aliasing layers, the mCE and FR results are normalized by those of the baseline network.

Network	**E**	**CE**	**mCE**	**FR**
ResNet-18	**30.24**	**68.32**	**1**	1
ResNet-18-pp	30.25	68.38	1.0024	1.069
ResNet-18-b1	30.98	68.98	1.0109	1.075
ResNet-18 + Rect-2	31.27	69.15	1.0146	0.899
ResNet-18 + Tri-3	31.33	68.94	1.0114	0.854
ResNet-18 + Bin-5	31.77	69.51	1.0198	**0.840**
ResNet-50	**23.84**	62.20	1	1
ResNet-50-pp	24.11	61.60	0.99	0.986
ResNet-50-b1	24.31	61.51	**0.989**	0.998
ResNet-50 + Rect-2	24.04	**61.34**	0.9919	0.886
ResNet-50 + Tri-3	24.13	61.40	0.9928	0.852
ResNet-50 + Bin-5	24.64	61.54	0.9950	**0.844**

CIFAR data set with respect to the low-pass filters, which overly smooth the intermediate response maps in the ConvNets. Models with anti-aliasing filters achieved less reduction of corruption error with respect to inhibition-augmented ConvNets: up to 10% on noise and around 2% on motion blur, but perform better $(> 6\%)$ than networks with inhibition on the translate perturbation.

Inhibition and data augmentation. We show that the embedding of the push-pull layer within existing architectures can be combined with data augmentation techniques to further improve the model robustness to different corruptions. In Table 2, we report the robustness results (CE and FR) of WideResNet models (with and without push-pull inhibition) trained with data augmentation techniques. We performed experiments with AutoAugment, cutout and a combination of them. The cutout augmentation does not increase the robustness of the models to input corruptions, while AutoAugment contributes to a noticeable reduction of the corruption error. This is in line with the findings by Yin et al. (2019). In all cases, inhibition layers contribute to further increase the robustness of networks trained with data augmentation. The best CE and FR values are obtained by models that deploy the push-pull layer in the first layer and in the first residual block, and are trained using AutoAugment plus cutout augmentations.

5.2 IMAGENET

In Table 3, we show the results achieved by ResNet models on the ImageNet validation set and on ImageNet-C/P benchmarks. The mCE and FR values are normalized with respect to the results of the baseline models ResNet-18 and ResNet-50, respectively.

On one hand, we observed a general decrease of performance of fine-tuned ResNet-18 models with push-pull and anti-aliasing layers on the ImageNet-C data set. On the other hand, the push-pull layer improves the robustness of ResNet-50 on the corruptions in ImageNet-C and on the perturbations in ImageNet-P. Results of the ResNet models extended with anti-aliasing layers decrease on the ImageNet-C data set with respect to those of the original ResNet model, while showing improved robustness to the perturbation of the ImageNet-P data set. Also for the ImageNet-C/P benchmarks, we noticed that the improvements of robustness achieved by the models with push-pull inhibition are directed towards high-frequency corruptions. Nonetheless, the complementary robustness shown by ResNet-50 architectures with push-pull inhibition and anti-aliasing layers suggest that they can be jointly deployed in existing models, as we also observed in the CIFAR experiments.

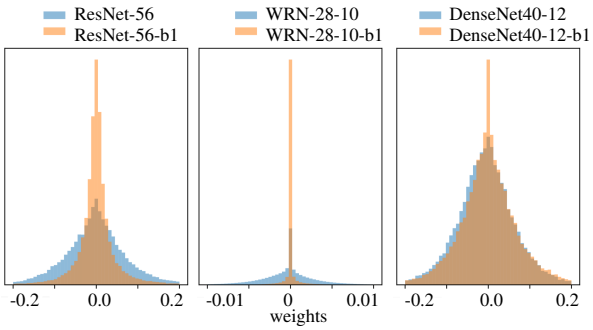

Figure 4: Histograms of the weights learned in push-pull and convolutional layers of (a) ResNet, (b) WideResNet and (c) DenseNet. Networks with push-pull layers (orange) learn sparser representations than networks with only convolutional layers (blue).

5.3 Discussion

The push-pull inhibition component that we implemented into ConvNets allows for a sharper detection of features of interest in cases when the input signal is corrupted. In neuroscience studies, it was found that while neurons that exhibit push-pull inhibition achieve similar activations to preferred stimuli when compared to neurons that do not exhibit such inhibition, the fact that their activations are suppressed in noisy areas result in a more pronounced distinction between patterns of interest from others and in a sparser representation (Li et al., 2012). These may be considered as two desirable properties in machine learning models. In Figure 4, we show the histograms of the weights learned in convolutional and push-pull layers of several models. The push-pull layer acts as a regularizer and allows to learn generally sparser representations. We conjecture that one effect of the push-pull layer is the learning of smoother discriminant mapping functions due to sparser representations. This contributes to the increased generalization showed on corrupted data (Srivastava et al., 2014). While the resulting models may fit slightly less than models without push-pull inhibition on training data, they generalize better to unknown corruptions and perturbations.

The push-pull layer contributes to the improvement of the robustness of a particular architecture to input corruptions, with negligible reduction of classification accuracy on the original data. When classification systems are deployed in real-world tasks, where the acquisition of data is subject to corruptions due to sensor uncertainty/deterioration or to changing environments, it is desirable to provide them with mechanisms that can effectively deal with such degradation. From a computational cost point of view, during the training stage, the pull kernels are derived at each iteration by upsampling and negating the corresponding push kernels, which relatively slows down the training process. On a Nvidia V100 gpu, one training iteration with a batch of 256 ImageNet images takes 0.78, 0.85 and 0.95 seconds for ResNet50, ResNet50-pp and ResNet50-b1, respectively. In the testing stage, the pull kernels are loaded only once when the model is initiated. Extra computations are due only to the processing of the pull component. When the push-pull layer is used only in the first layer of the network or in the first block, the difference in computations is negligible. For inference, the processing takes 0.66, 0.67 and 0.63 seconds, respectively. The lowest test time of ResNet50-b1 is attributable to the sparser representation learned in the push-pull layer.

6 Conclusions

We demonstrated that the inclusion of a response inhibition mechanism, inspired by the push-pull neurons of the visual system of the brain, into existing ConvNets improves their robustness to corruptions and perturbations of the input that are not present in the training data. The improvement is mainly attributable to the filters in the push-pull layer that are less sensitive to noise, and have a regularization effect that makes the network learn a sparser representation. We carried out experiments on the CIFAR-C/P and ImageNet-C/P data sets and the results that we achieved (more than 10% of reduction of the error on corrupted and perturbed data, and 1% of reduction of the corruption error for fine-tuned networks) demonstrate the effectiveness of the push-pull inhibition layer.

REFERENCES

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. doi: 10.1109/access.2018.2807385.

George Azzopardi, Antonio Rodríguez-Sánchez, Justus Piater, and Nicolai Petkov. A push-pull corf model of a simple cell with antiphase inhibition improves snr and contour detection. *PLoS ONE*, 9(7):e98424, 07 2014. doi: 10.1371/journal.pone.0098424.

Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In *NIPS*, pp. 7538–7550. 2018.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Judith A. Hirsch, José-Manuel Alonso, R. Clay Reid, and Luis M. Martinez. Synaptic integration in striate cortical simple cells. *Journal of Neuroscience*, 18(22):9517–9528, 1998. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.18-22-09517.1998.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, pp. 2261–2269, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1097–1105. 2012.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

Thomas Z. Lauritzen and Kenneth D. Miller. Different roles for simple-cell and complex-cell inhibition in v1. *Journal of Neuroscience*, 23(32):10201–10213, 2003. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.23-32-10201.2003.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *AISTATS*, volume 38, pp. 562–570, 2015.

Ya-tang Li, Wen-pei Ma, Ling-yun Li, Leena A. Ibrahim, Sheng-zhi Wang, and Huizhong Whit Tao. Broadening of inhibitory tuning underlies contrast-dependent sharpening of orientation selectivity in mouse visual cortex. *Journal of Neuroscience*, 32(46):16466–16477, 2012. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.3221-12.2012.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *CoRR*, abs/1905.00397, 2019.

Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D. Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *CoRR*, abs/1906.02611, 2019.

Jiajun Lu, Hussein Sibai, Evan Fabry, and David A. Forsyth. Standard detectors aren't (currently) fooled by physical adversarial stop signs. *CoRR*, abs/1710.03337, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2018.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *ICLR*, 2017. URL https://arxiv.org/abs/1702.04267.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, pp. 86–94, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Juan C. Pérez, Motasem Alfarra, Guillaume Jeanneret, Adel Bibi, Ali Thabet, Bernard Ghanem, and Pablo Arbeláez. Gabor layers enhance network robustness. *arXiv:1912.05661*, 2020.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

N. Strisciuglio, G. Azzopardi, and N. Petkov. Robust inhibition-augmented operator for delineation of curvilinear structures. *IEEE Trans Image Process*, 28(12):5852–5866, 2019. doi: 10.1109/TIP.2019.2922096.

Nicola Strisciuglio, Manuel Lopez-Antequera, and Nicolai Petkov. Enhanced robustness of convolutional networks with a push–pull inhibition layer. *Neural Computing and Applications*, 2020. doi: 10.1007/s00521-020-04751-8.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *CoRR*, abs/1611.05760, 2016. URL http://arxiv.org/abs/1611.05760.

Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D. Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *CoRR*, abs/1906.08988, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.

S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *CVPR*, pp. 4480–4488, June 2016. doi: 10.1109/CVPR.2016.485.

APPENDIX

A PUSH-PULL RESIDUAL AND DENSE LAYERS

In Figure 5, we show the structure of the composite residual and dense layers augmented with the push-pull inhibition. The architecture of the layers is the same of the original residual and dense layers, with the difference that all convolutions are replaced by push-pull filters.

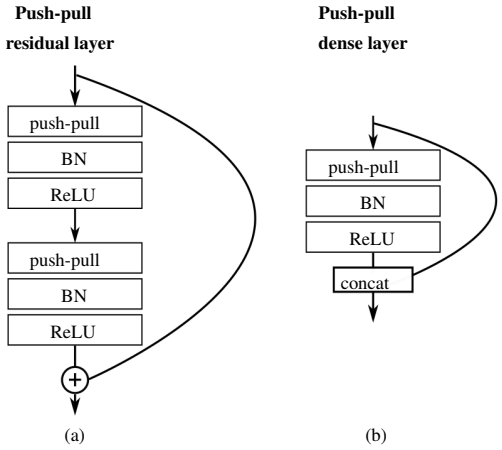

Figure 5: Modified residual and dense layers. In the (a) push-pull residual layer and (b) push-pull dense layer we replace the convolutions with our push-pull layer.

B PERFORMANCE METRICS

We denote by E^M the classification error of a model M on the original clean (without perturbations) test set. Let us denote by $E^M_{c,s}$ the corruption error achieved by the model M on images with corruption $c \in C$ and severity level s, and denote by E^M_c the average error achieved across all severity sets of a given corruption c.

We compute the mean corruption error mCE as the average of the normalized errors achieved on all corruptions and severity levels. We use the error $E^{baseline}_{c,s}$ achieved by a baseline network on the set of data with corruption c and severity s as a normalization factor for the corruption error:

$$mCE^M = \frac{1}{|C|}\sum_{c=1}^{|C|}\frac{\sum_{s=1}^{5} E^M_{c,s}}{\sum_{s=1}^{5} E^{baseline}_{c,s}} \tag{2}$$

As proposed by Hendrycks & Dietterich (2019), we also evaluate the degradation of the classifier performance on corrupted data with respect to the results achieved on clean data and compute the relative mean corruption error rCE. It measures the relative gap between the error on clean and corrupted data and it is computed as:

$$rCE^M = \frac{1}{|C|}\sum_{c=1}^{|C|}\frac{\sum_{s=1}^{5}\left(E^M_{c,s} - E^M\right)}{\sum_{s=1}^{5}\left(E^{baseline}_{c,s} - E^{baseline}\right)} \tag{3}$$

As a measure of performance on the CIFAR-P and ImageNet-P test sets, we compute the flip probability FP. It measures the probability that the outcome of the classification changes between consecutive frames of a sequence S due to a perturbation p. Formally, it is defined as:

$$FP^M_p = \mathbb{P}_{x\sim S}(f(_j) \neq f(x_{j-1})) \tag{4}$$

where x_j is the j-th frame of the perturbation sequence. Similarly to the mCE, the flip rate $FR^M_p = FP^M_p / FP^{baseline}_p$ is the normalized version of the FP.

C DETAILED RESULTS

C.1 RESULTS PER CORRUPTION TYPE

In Table 4, we show the detailed results achieved on the corruption sets in the CIFAR-C data set by network with and without push-pull inhibition. We also report the results achieved by networks that deploy the anti-aliased sampling filters of Zhang (2019). The push-pull filters are robust to high-frequency corruptions and can be effectively combined with anti-aliasing filters.

Table 4: Results on CIFAR-C, for models that make a combined use of the push-pull filters with the anti-aliasing filters.

			noise			blur			
	lpf	**original**	**Gaussian**	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**
AlexNet		13.87	38.44	31.82	46.13	23.53	40.97	29.32	26.89
ResNet-20		7.56	57.41	45.75	41.88	21.43	58.82	30.62	27.54
ResNet-20	2	7.37	57.3	46.06	43.39	22.13	54.95	27.89	29.56
ResNet-20	3	7.93	56.39	45.36	43.14	22.72	61.53	29.5	30.02
ResNet-20	5	7.98	60.44	48.81	45.49	23.36	59.71	30.97	32.43
ResNet-20-pp		8.29	49.90	39.98	37.20	22.30	58.56	29.50	29.68
ResNet-20-pp	2	7.95	48.58	39.61	35.39	21.87	56.93	29.18	28.11
ResNet-20-pp	3	7.78	45.76	36.84	34.63	20.9	54.47	28.28	27.13
ResNet-20-pp	5	8.45	46.37	37.08	35.72	23.61	55.13	29.98	31.09
ResNet-20-b1		8.32	47.60	37.31	38.05	22.31	51.64	29.76	28.95
ResNet-20-b1	2	8.2	45.31	36.41	35.85	21.87	53.89	27.79	26.99
ResNet-20-b1	3	8.45	45.28	35.84	35.92	21.74	56.43	29.92	28.38
ResNet-20-b1	5	7.93	47.6	37.99	37.64	19.15	54.83	27.3	24.17
ResNet-56		6.64	56.72	44.02	44.68	20.02	59.22	26.54	26.73
ResNet-56	2	6.42	49.28	38.78	38.39	20.18	55.79	25.85	26.64
ResNet-56	3	6.28	53.54	42.04	40.44	20.91	58.11	28.56	29.14
ResNet-56	5	6.56	54.72	43.25	40.72	20.97	55.55	27.35	27.72
ResNet-56-pp		7.01	44.92	35.74	33.14	20.88	50.46	27.67	26.58
ResNet-56-pp	2	7.11	41.43	32.99	32.62	18.81	44.6	25.12	23.7
ResNet-56-pp	3	6.86	44.33	35.36	33.21	21.09	54.04	27.70	28.53
ResNet-56-pp	5	6.7	44.8	34.87	32.43	21.2	55.65	28.83	29.06
ResNet-56-b1		7.76	40.17	31.88	29.56	18.56	45.53	25.36	23.36
ResNet-56-b1	2	10.87	27.34	23.15	35.31	14.09	17.19	18.78	15.33
ResNet-56-b1	3	7.37	38	29.4	33.59	19.1	45.28	25.01	23.84
ResNet-56-b1	5	8.45	31.91	26.8	30.99	17.1	30.42	20.58	19.81

			weather			digital			
	lpf	**snow**	**frost**	**fog**	**brightness**	**contrast**	**elastic**	**pixelate**	**jpeg**
AlexNet		26.94	28.26	30.31	17.84	46.77	22.25	18.36	18.48
ResNet-20		25.06	31.39	16.56	9.73	29.59	20.37	32.33	23.81
ResNet-20	2	25.18	31.04	15.71	9.7	28.19	20.66	34.87	24.22
ResNet-20	3	24.98	32.16	17.16	10.12	28.82	21.11	33.24	24.57
ResNet-20	5	25.61	34.78	17.36	10.25	28.75	21.55	35.83	25.27
ResNet-20-pp		25.59	31.29	17.58	10.34	30.50	20.65	32.70	22.16
ResNet-20-pp	2	25.3	29.89	16.94	9.99	29.88	20.72	31.3	21.69
ResNet-20-pp	3	24.64	27.39	15.75	10	26.15	20.52	31.7	22.37
ResNet-20-pp	5	26.36	30	17.06	10.55	28.96	21.28	34.02	22.16
ResNet-20-b1		24.00	26.92	18.04	10.29	31.15	19.56	27.95	20.35
ResNet-20-b1	2	23.79	28.69	16.59	10.45	29.24	19.33	31.28	20.58
ResNet-20-b1	3	25.89	29.59	17.99	10.48	28.06	21.14	33.9	22.14
ResNet-20-b1	5	24.37	28.65	15.99	10.16	28.68	18.57	32.37	21.28
ResNet-56		21.99	29.08	14.55	8.30	24.89	18.52	29.79	22.88
ResNet-56	2	21.94	28.12	15.73	8.28	24.44	17.81	31.31	22.43
ResNet-56	3	21.95	28.7	15.67	8.11	25.49	19.87	29.91	22.76
ResNet-56	5	22.78	28.81	15.57	8.33	24.58	19.56	31.26	22.89
ResNet-56-pp		22.32	27.28	15.93	8.68	26.94	18.57	29.06	19.82
ResNet-56-pp	2	21.19	24.74	16.52	9.17	28.9	16.84	23.28	17.49
ResNet-56-pp	3	22.52	25.94	15.70	8.85	25.81	19.34	28.50	21.03
ResNet-56-pp	5	22.01	27	15.28	8.62	25.3	19.05	28.77	19.79
ResNet-56-b1		22.51	25.49	16.31	9.78	29.30	17.95	25.94	17.73
ResNet-56-b1	2	18.94	17.67	23.73	13.39	34.84	15.68	12.32	15.89
ResNet-56-b1	3	20.87	24.31	16.5	9.22	27.91	16.76	25.18	16.89
ResNet-56-b1	5	18.14	19.42	19.5	10.56	32.47	15.95	14.08	15.72

In Table 5, we report detailed results per corruption type achieved by WideResNet models for which the convolutions in the first layer and in the first residual block were replaced by push-pull filters, and that are trained using different types of data augmentation, namely the cutout (Devries & Taylor,

2017) and AutoAugment (Cubuk et al., 2018) techniques. The use of the push-pull filters can be combined with data augmentation to further improve the robustness of existing models.

Table 5: Results on the corruptions in CIFAR-C, for models that make a combined use of the push-pull filters with cutout and AutoAugment (AA) data augmentation techniques.

			noise						
	augmentation	**original**	**Gaussian**	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**
AlexNet		13.87	38.44	31.82	46.13	23.53	40.97	29.32	26.89
WRN-28-10		3.79	51.25	38.66	40.08	15.55	43.29	19.73	20.17
WRN-28-10-pp		4.09	35.46	26.36	29.14	15.52	40.2	18.91	19.17
WRN-28-10-b1		4.59	37.54	28.21	28.61	15.02	41.91	20.26	17.5
WRN-28-10	AA	2.84	36.95	26.53	15.71	5.2	35.57	10.16	6.44
WRN-28-10-pp	AA	3.37	31.82	22.74	16.73	5.94	35.61	12.86	7.54
WRN-28-10-b1	AA	3.2	27.04	19.24	16.3	6.09	36.73	12.08	7.94
WRN-28-10	cutout	3.18	56.64	45.24	50.99	17.83	38.03	19.41	25
WRN-28-10-pp	cutout	3.32	56.07	44.09	45.24	17.15	46.78	19.38	21.45
WRN-28-10-b1	cutout	3.43	45.46	34.84	40.32	19.46	41.63	18.73	24.02
WRN-28-10	AA+cutout	2.67	38.83	27.75	13.53	5.8	31.95	11.66	7.01
WRN-28-10-pp	AA+cutout	2.71	39.47	28.18	16.85	5.66	39.57	11.83	7.23
WRN-28-10-b1	AA+cutout	2.9	33.08	23.39	17.28	5.91	33.2	11.73	7.44

			weather			digital			
	augmentation	**snow**	**frost**	**fog**	**brightness**	**contrast**	**elastic**	**pixelate**	**jpeg**
AlexNet		26.94	28.26	30.31	17.84	46.77	22.25	18.36	18.48
Wrn-28-10		14.74	18.72	9.84	5.12	18.37	14.56	24.57	21.81
Wrn-28-10-pp		15.75	17.71	10.92	5.44	18.32	13.6	20.59	17.71
Wrn-28-10-b1		16.27	19.14	11.77	5.95	21.92	13.54	19.59	16.86
Wrn-28-10	AA	10.07	11.68	5.25	3.28	4.75	9.99	25.27	17.68
Wrn-28-10-pp	AA	11.67	13.44	7.31	3.88	7.01	10.77	23.63	14.39
Wrn-28-10-b1	AA	12.14	12.83	7.08	3.81	6.02	10.64	23.26	13.55
Wrn-28-10	cutout	12.33	16.53	8.9	4.51	15.75	12.75	23.61	21.07
Wrn-28-10-pp	cutout	14.94	20.7	9.54	4.66	15.98	11.62	22.3	17.69
Wrn-28-10-b1	cutout	13.67	17.3	9.35	4.69	15.07	11.27	19.28	15.35
Wrn-28-10	AA+cutout	8.64	11.46	5.26	3.21	5	9.6	28.94	18.02
Wrn-28-10-pp	AA+cutout	11.44	15.24	6.57	3.26	6.78	9.28	23.78	15.45
Wrn-28-10-b1	AA+cutout	10.01	12.46	6.3	3.39	5.32	8.98	20.37	13.12

In Table 6, we report the detailed classification error achieved per corruption type on ImageNet-C by models that deploy the push-pull layers in comparison with those achieved by models that use the antialiased sampling module proposed by Zhang (2019). The push-pull layers provide robustness for high-frequency corruptions, that is more evident in the case of the ResNet-50 models.

Table 6: Detailed comparison of results on the corruptions in ImageNet-C achieved by models that deploy push-pull and anti-aliasing-filters.

			noise			blur			
	lpf	**original**	**Gaussian**	**shot**	**impulse**	**defocus**	**glass**	**motion**	**zoom**
Resnet-18		30.24	77.33	79.19	82.43	71.97	77.03	70.38	70.43
ResNet-18-pp		30.25	76.41	77.74	81.6	72.48	75.72	69.7	70.47
ResNet-18-b1		30.98	77.73	79.31	83.29	73.02	75.24	69.61	70.84
Resnet-18	2	31.27	77.65	79.55	83.31	71.58	76.51	69.14	70.43
Resnet-18	3	31.33	77.09	79.08	82.46	71.83	76.48	69.41	70.98
Resnet-18	5	31.77	78.00	79.94	83.43	72.26	76.65	69.55	71.45
ResNet-50		23.84	68.37	70.5	72.87	64.32	74.32	63.11	64.83
ResNet-50-pp		24.11	66.99	69.02	71.88	64.92	72.65	63.77	64.25
ResNet-50-b1		24.31	67.57	70.13	73.09	64.59	72.32	63	63.66
ResNet-50	2	24.04	67.62	70.13	72.15	63.60	73.75	62.54	65.14
ResNet-50	3	24.13	67.49	69.74	71.73	62.97	74.12	62.71	64.82
ResNet-50	5	24.64	67.68	70.00	72.06	63.57	73.99	62.68	64.95

			weather			digital			
	lpf	**snow**	**frost**	**fog**	**brightness**	**contrast**	**elastic**	**pixelate**	**jpeg**
Resnet-18		75.9	71.78	66.12	41.15	69.2	60.22	57.87	53.75
ResNet-18-pp		75.89	71.48	67.5	42.16	71.67	60.08	57.95	54.8
ResNet-18-b1		76.45	72.19	68.66	42.95	73.29	59.54	58.35	54.25
Resnet-18	2	76.28	72.24	68.15	43.07	71.41	59.94	62.52	55.54
Resnet-18	3	75.79	71.96	67.63	42.78	71.11	59.86	62.15	55.42
Resnet-18	5	76.86	72.98	68.28	43.36	71.55	60.20	62.55	55.60
ResNet-50		70.35	65.66	59.44	35.81	64.34	56.68	55.18	47.16
ResNet-50-pp		70.53	64.85	60.24	35.15	64.8	55.6	52.62	46.71
ResNet-50-b1		70.41	64.43	58.97	35.5	64.25	55.46	51.49	47.8
ResNet-50	2	69.03	64.39	57.13	35.22	62.92	56.21	52.67	47.55
ResNet-50	3	69.72	65.05	57.08	35.22	63.28	56.12	53.99	46.94
ResNet-50	5	69.96	65.79	57.73	35.60	63.12	56.68	53.00	46.29

C.2 RESULTS PER PERTURBATION TYPE

We report in Table 7 the detailed performance results, in terms of flip probability, per perturbation type achieved on the CIFAR-P data set by ResNet models, of different depth, that deploy the push-pull layers, the anti-aliasing layers or a combination of them. The flip probability measures the likelihood that a model changes its predictions on consecutive frames of a video on which an incremental corruption is applied. The push-pull inhibition layers contribute to a substantial improvement of stability against high-frequency components, and benefit from the effect of the anti-aliasing filters on the translate perturbation.

Table 7: Flip probability per perturbation achieved on the CIFAR-P data set by networks augmented with push-pull filters and anti-aliasing filters.

		noise		blur		weather		digital			
	lpf	**Gaussian**	**shot**	**motion**	**zoom**	**snow**	**bright.**	**translate**	**rotate**	**tilt**	**scale**
AlexNet		0.041	0.051	0.084	0.006	0.036	0.016	0.136	0.089	0.033	0.105
ResNet-20		0.082	0.111	0.119	0.008	0.037	0.009	0.046	0.053	0.018	0.065
ResNet-20	2	0.080	0.108	0.118	0.008	0.038	0.010	0.044	0.054	0.019	0.064
ResNet-20	3	0.077	0.102	0.118	0.007	0.040	0.009	0.042	0.055	0.019	0.067
ResNet-20	5	0.083	0.106	0.111	0.008	0.037	0.010	0.042	0.053	0.018	0.065
ResNet-20-pp		0.076	0.097	0.123	0.008	0.039	0.014	0.053	0.055	0.019	0.067
ResNet-20-pp	2	0.070	0.089	0.122	0.007	0.038	0.013	0.048	0.055	0.019	0.070
ResNet-20-pp	3	0.074	0.096	0.120	0.008	0.037	0.014	0.046	0.055	0.018	0.066
ResNet-20-pp	5	0.072	0.092	0.120	0.009	0.039	0.012	0.043	0.057	0.020	0.069
ResNet-20-b1		0.062	0.081	0.114	0.007	0.034	0.014	0.053	0.051	0.018	0.066
ResNet-20-b1	2	0.061	0.078	0.108	0.007	0.034	0.013	0.052	0.054	0.018	0.067
ResNet-20-b1	3	0.063	0.080	0.103	0.006	0.033	0.013	0.048	0.048	0.019	0.066
ResNet-20-b1	5	0.067	0.085	0.110	0.007	0.035	0.014	0.049	0.053	0.018	0.069
ResNet-56		0.0706	0.0943	0.1048	0.0063	0.0333	0.0076	0.0354	0.0458	0.0139	0.0523
ResNet-56	2	0.0656	0.0865	0.1021	0.0063	0.0303	0.0078	0.0329	0.0462	0.0143	0.0567
ResNet-56	3	0.0731	0.0987	0.0980	0.0063	0.0324	0.0079	0.0326	0.0451	0.0137	0.0549
ResNet-56	5	0.0688	0.0968	0.1096	0.0065	0.0344	0.0077	0.0348	0.0469	0.0152	0.0569
ResNet-56-pp		0.0577	0.0762	0.1080	0.0059	0.0304	0.0106	0.0365	0.0465	0.0147	0.0579
ResNet-56-pp	2	0.0644	0.0813	0.1105	0.0066	0.0329	0.0122	0.0378	0.0497	0.0162	0.0586
ResNet-56-pp	3	0.0573	0.0736	0.0979	0.0060	0.0327	0.0121	0.0381	0.0418	0.0143	0.0537
ResNet-56-pp	5	0.0615	0.0765	0.1098	0.0067	0.0318	0.0126	0.0382	0.0483	0.0157	0.0568
ResNet-56-b1		0.0517	0.0654	0.1007	0.0061	0.0297	0.0130	0.0429	0.0438	0.0158	0.0573
ResNet-56-b1	2	0.0373	0.0485	0.0814	0.0044	0.0219	0.0114	0.0494	0.0393	0.0137	0.0530
ResNet-56-b1	3	0.0540	0.0702	0.0959	0.0055	0.0305	0.0117	0.0379	0.0482	0.0151	0.0593
ResNet-56-b1	5	0.0514	0.0667	0.1009	0.0056	0.0291	0.0111	0.0383	0.0490	0.0154	0.0582

In Table 8 we report detailed flip probability results per perturbation type achieved by WideResNet models that deploy push-pull filters in the first layer and in the first residual block to replace the original convolutions, and that are trained using different types of data augmentation, namely the cutout (Devries & Taylor, 2017) and AutoAugment (Cubuk et al., 2018) techniques. The push-pull inhibition and the different data augmentation techniques provide complementary contributions to the improvement of the model robustness and stability to perturbations. This demonstrates that the push-pull layer can be deployed in existing architectures and can be combined with other approaches to further improve the classification performance.

Table 8: Flip probability per perturbation achieved on the CIFAR-P data set by networks augmented with push-pull filters and trained with different data augmentation techniques. AA stands for AutoAugment.

		noise		blur		weather		digital			
	augm.	**Gaussian**	**shot**	**motion**	**zoom**	**snow**	**brightness**	**translate**	**rotate**	**tilt**	**scale**
AlexNet		0.0411	0.0507	0.0842	0.0059	0.0364	0.0160	0.1364	0.0894	0.0326	0.1050
WRN-28-10		0.0479	0.0693	0.0815	0.0039	0.0205	0.0049	0.0186	0.0276	0.0075	0.0334
WRN-28-10-pp		0.0387	0.0524	0.0787	0.0034	0.0204	0.0076	0.0207	0.0273	0.0080	0.0334
WRN-28-10-b1		0.0405	0.0531	0.0771	0.0033	0.0218	0.0081	0.0231	0.0280	0.0087	0.0370
WRN-28-10	AA	0.0345	0.0464	0.0485	0.0029	0.0133	0.0035	0.0145	0.0171	0.0062	0.0202
WRN-28-10-pp	AA	0.0351	0.0431	0.0560	0.0029	0.0151	0.0066	0.0166	0.0201	0.0065	0.0235
WRN-28-10-b1	AA	0.0312	0.0383	0.0560	0.0028	0.0162	0.0056	0.0174	0.0217	0.0074	0.0238
WRN-28-10	cutout	0.0481	0.0729	0.0906	0.0044	0.0179	0.0046	0.0140	0.0265	0.0069	0.0304
WRN-28-10-pp	cutout	0.0435	0.0622	0.0821	0.0035	0.0213	0.0062	0.0161	0.0225	0.0072	0.0263
WRN-28-10-b1	cutout	0.0390	0.0516	0.0802	0.0039	0.0186	0.0058	0.0162	0.0215	0.0075	0.0269
WRN-28-10	AA+cutout	0.0308	0.0429	0.0577	0.0028	0.0120	0.0033	0.0122	0.0163	0.0058	0.0179
WRN-28-10-pp	AA+cutout	0.0341	0.0441	0.0535	0.0027	0.0147	0.0048	0.0148	0.0174	0.0063	0.0186
WRN-28-10-b1	AA+cutout	0.0283	0.0377	0.0499	0.0029	0.0131	0.0046	0.0157	0.0180	0.0062	0.0185

In Table 9 we report the detailed flip probability results achieved by ResNet models augmented with push-pull inhibition filters in comparison to those achieved by models that deploy anti-aliasing layers. The push-pull filters, especially in the case of ResNet-50, contribute to the improvement of the stability of the network prediction on consecutively perturbed frames and are particularly effective on high-frequency corruptions.

Table 9: Flip probability results per perturbation achieved on the ImageNet-P data set by ResNet models augmented with push-pull filters, compared with those achieved by ResNet models that deploy anti-aliasing filters.

		noise		**blur**		**weather**		**digital**			
	augm.	**Gaussian**	**shot**	**motion**	**zoom**	**snow**	**brightness**	**translate**	**rotate**	**tilt**	**scale**
Resnet-18		0.0979	0.1154	0.0700	0.0504	0.0901	0.0380	0.0689	0.0869	0.0517	0.1560
ResNet-18-pp		0.1009	0.1213	0.0736	0.0527	0.0927	0.0396	0.0797	0.0964	0.0575	0.1662
ResNet-18-b1		0.0968	0.1156	0.0736	0.0518	0.0958	0.0389	0.0837	0.0983	0.0594	0.1710
Resnet-18	2	0.0893	0.1052	0.0661	0.0473	0.0812	0.0375	0.0499	0.0752	0.0440	0.1479
Resnet-18	3	0.0844	0.1026	0.0631	0.0466	0.0777	0.0370	0.0429	0.0689	0.0420	0.1394
Resnet-18	5	0.0837	0.1007	0.0632	0.0467	0.0798	0.0369	0.0376	0.0662	0.0416	0.1355
Resnet-50		0.0748	0.0879	0.0603	0.0437	0.0832	0.0324	0.0486	0.0702	0.0413	0.1149
Resnet-50-pp		0.0717	0.0847	0.0591	0.0416	0.0783	0.0331	0.0527	0.0692	0.0402	0.1148
Resnet-50-b1		0.0715	0.0841	0.0605	0.0421	0.0795	0.0335	0.0538	0.0703	0.0404	0.1171
Resnet-50	2	0.0666	0.0778	0.0554	0.0404	0.0686	0.0300	0.0350	0.0581	0.0346	0.1034
Resnet-50	3	0.0642	0.0756	0.0538	0.0399	0.0692	0.0300	0.0300	0.0540	0.0330	0.0977
Resnet-50	5	0.0648	0.0762	0.0534	0.0397	0.0707	0.0297	0.0270	0.0527	0.0333	0.0957