Visual response inhibition for increased robustness of convolutional networks to distribution shifts

Nicola Strisciuglio[1] **George Azzopardi**[2]
[1]University of Twente, Enschede, The Netherlands
[2]University of Groningen, Groningen, The Netherlands
n.strisciuglio@utwente.nl, g.azzopardi@rug.nl

Abstract

Convolutional neural networks have been shown to suffer from distribution shifts in the test data, for instance caused by the so called common corruptions and perturbations. Test images can contain noise, digital transformations, and blur that were not present in the training data, negatively impacting the performance of trained models. Humans experience much stronger robustness to noise and visual distortions than deep networks. In this work, we explore the effectiveness of a neuronal response inhibition mechanism, called push-pull, observed in the early part of the visual system, to increase the robustness of deep convolutional networks. We deploy a Push-Pull inhibition layer as a replacement of the initial convolutional layers (input layer and in the first block of residual and dense architectures) of standard convolutional networks for image classification. We show that the Push-Pull inhibition component increases the robustness of standard networks for image classification to distribution shifts on the CIFAR10-C and CIFAR10-P test sets.

1 Introduction

Despite exceptional results and over-human performance achieved on benchmark visual processing tasks, f.i. image classification (1; 2; 3; 4), convolutional networks have shown decrease of performance when the test images are affected by unexpected changes (5). The nature of these changes can be artificial, resulting in subtle and imperceptible adversarial attacks (6; 7; 8; 9), or natural, determined by common corruptions and perturbations in computer vision, such as noise, contrast changes, elastic transformations, blur, and so on (10). The lack of robustness has been attributed to the tendency of the network to learn superficial statistics that exist in the training data (11; 12) that favour 'easy' solutions to the optimization problem, namely a simplicity bias (13). This is also regarded as shortcut learning (14).

One common approach to increase the generalization and robustness to distribution shifts of convolutional networks is data augmentation, in which models are trained by extending the training data with (patch-wise) noise (15) or blur(16), masking random patches (17), or with more advanced schemes such as AutoAugment (18; 19), AugMix (20) or DeepAugment (21), among others. Data augmentation is a powerful tool, but can cause underfitting (5) or a misleading sense of robustness. For example, using high-frequency noise for data augmentation (e.g. Gaussian noise) makes the network focusing on low frequency features that are discriminant instead of becoming explicitly robust to high-frequency corruptions (22). In (22), it was suggested that architectural changes to networks, and robustness by design are promisng directions to undertake, and eventually be combined with data augmentation.

The human visual system, instead, has remarkably strong generalization capabilities across a large range of changes of the input distribution, e.g. illumination changes, blur and motion-related distortion, noise, and weather conditions (5), among others. Part of the robustness of the visual

Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

system has been attributed to neural response inhibition mechanisms (23), e.g. the push-pull inhibition that is exhibited by simple neurons and acts as a spatial frequency tuning function to increase the stability of cortical activity (24). A simple neuron that exhibits push-pull inhibition is believed to be connected with an interneuron whose activity suppresses (inhibits) the activity of the neuron concerned. An inhibitory interneuron responds to the same preferred stimulus, but of *opposite contrast*, of the neuron to which it is connected. For simplification sake, we can think of a push-pull neuron as having two receptive fields, referred to as the excitatory (push) and inhibitory (pull). Both receptive fields are similar in structure but are in antiphase of each other, with the inhibitory receptive field tending to be larger (25). A stimulus that elicits the activation of the interneuron may suppress, to some extent, the response of the excitatory one. A computational model of the push-pull inhibition was introduced in image processing filters with application to contour and line (26; 27; 28) detection that demonstrated its outstanding performance in terms of robustness against various types of noise.

In this work, we deploy a push-pull layer (29) as a substitute of the classical convolutional layer in CNNs and investigate its impact on improving the robustness of image classification models to distribution shifts caused by common corruptions and perturbations using the CIFAR10-C and CIFAR10-P benchmarks (10). We build on (29), and here we deploy the push-pull layer to replace all convolutions up to the first block of residual and dense networks. Furthermore, we combine these modifications with data augmentation strategies to train networks, demonstrating their complementarity to further boost robustness to distribution shifts caused by visual corruptions and perturbations.

2 Push-Pull inhibition layer

The response of a push-pull layer (Figure 1) is the combination of the activation map of two convolutions processed with a non-linear function (ReLU, in our case), one with an excitatory (push) kernel $k(\cdot)$ and one with an inhibitory (pull) kernel $-\hat{k}(\cdot)$ (29). The response R of a push-pull layer is computed as:

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

where $\Theta(\cdot)$ is the ReLU function, α is the inhibition strength, 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. The pull kernel has a larger receptive field, to allow for suppression of the inhibitory response on the pattern of interest and in the immediate surrounding, as found in neurophysiological studies (30). In the experiments, the value α and h were set to 1 and 2, respectively, following indications in (29) and a grid search. Figure 2 shows an example of the application of one push-pull operation. An implementation of the push-pull layer is publicly available[1].

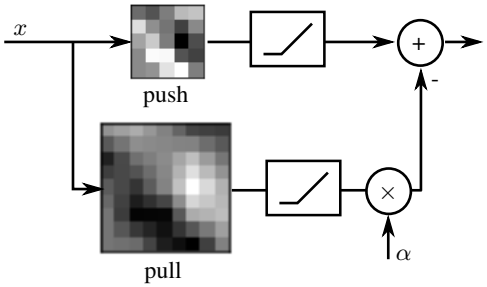

Figure 1: Structure of the push-pull layer. An input image x is convolved with a push and a pull kernel. The pull kernel is derived from the push one by negation and upsampling (by a factor h). Their responses are rectified and combined linearly.

2.1 Push-pull in residual and dense blocks

We extend the residual and dense composite layers (1; 3) by substituting the convolutional layers with push-pull layers. This allows us to investigate the affect of the inhibition component up to the first block of ResNet and DenseNet architectures with respect to the robustness against distribution shifts. The design decision to deploy the push-pull inhibition up to the fist block is inspired by neurophysiological studies that observed this kind of inhibition exihibited in area V1 of the visual cortex (24). In the Appendix, we report a scheme of the modified residual and dense layers.

3 Experiments and results

Training and evaluation. We trained several ResNet and DenseNet models and their versions with the push-pull layer, using the CIFAR10 training set to which we applied a light data augmentation,

[1]Code available at https://github.com/nicstrisc/Push-Pull-CNN-layer

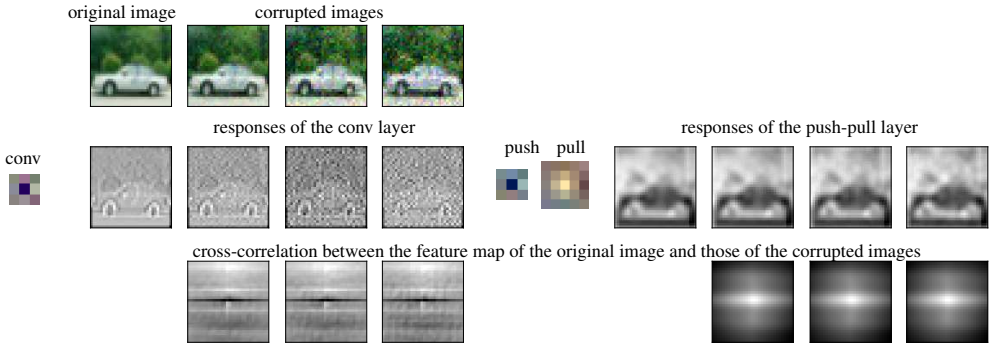

Figure 2: Robustness to noise: (top left) an image from CIFAR-10 and its corrupted versions with Gaussian noise of increasing severity; (middle left) the response maps of the convolutional kernel is affected by noise, while (right) the response of the push-pull kernel are less sensitive to it.

Table 1: Results obtained on the CIFAR10-C and CIFAR10-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 CIFAR10(-C/P)

Network	**E**	**mCE**	**rCE**	**mFR**
AlexNet (baseline)	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-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**

consisting only of center-cropping and random horizontal flipping (31). For each architecture, we trained the original model with convolutional layers and two other models, one (-pp) with push-pull 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. Training details are reported in the Appendix.

We applied the evaluation protocol proposed in (10), testing the models also on the corruption CIFAR10-C dataset and the perturbation CIFAR10-P dataset. Besides the classification error, we compute the corruption error C, i.e. the classification error achieved on the CIFAR10-C set, and the flip-probability FP, that is the probability that the outcome of the classification changes between consecutive frames, on the sequences in the CIFAR10-P set. We compute the mean and relative corruption errors mCE and rCE and the normalized flip probability, i.e. the flip rate FR. Details about these metrics are in (10).

Robustness to corruptions and perturbations. In Table 1, we report the results of ResNet and DenseNet networks on the CIFAR10 and CIFAR10-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). The measurements mCE, rCE and mFR are normalized with respect to the results of the baseline AlexNet.

The push-pull layer contributes to the improvement of the robustness of an existing model to distribution shifts caused by corruptions or perturbations of the input images, with negligible reduction

Table 2: Mean corruption error (mCE) and flip rate (FR) of WideResNet models, normalized wrt the AlexNet baseline, 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
	WRN-28-10-pp	0.689	0.521	0.804	0.548
	WRN-28-10-b1	0.703	0.499	0.728	**0.484**
FR	WRN-28-10	0.602	0.408	0.611	0.391
	WRN-28-10-pp	0.555	0.445	0.567	0.417
	WRN-28-10-b1	0.572	0.428	0.529	**0.384**

of the classification accuracy on the original data. Its effect, especially when it is deployed in the first block of existing convolutional networks, is noticeable, with a reduction of the mean corruption error mCE by more than 10% with respect to the counterpart network without inhibition, 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 a substantial 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 generally 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 CIFAR10-P sequences, also improved for all models with push-pull layers. We report detailed results per corruption type in the Appendix.

Networks with the push-pull layers generally show an improvement of robustness to input 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). We include the detailed results per corruption and perturbation type in the Appendix.

Push-pull inhibition and data augmentation. We also trained WideResNet models (2) with and without the push-pull layer, using different data augmentations, namely cutout (17), AutoAugment (18), and a combination of cutout and AutoAugment.

We show that the push-pull layer can be combined with the use of data augmentation techniques to further improve the model robustness to different corruptions and perturbations of test images. In Table 2, we report the robustness results (mCE and FR) of WideResNet models (with and without push-pull inhibition) trained with data augmentation techniques. 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 in (22). In all cases, push-pull inhibition layers contribute to further increase the robustness of networks trained with data augmentation. The best mCE 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. In the Appendix, we report detailed results per type of corruption and perturbation.

4 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 convolutional networks 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 perturbations, especially at high frequency.We carried out experiments on the CIFAR-C/P datasets and the results that we achieved (more than 10% of reduction of the error on corrupted and perturbed data) demonstrate the effectiveness of the push-pull inhibition layer.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[2] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[3] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *CVPR*, pp. 2261–2269, 2017.

[4] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR*, vol. abs/1611.05431, 2016.

[5] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, "Generalisation in humans and deep neural networks," in *NIPS*, pp. 7538–7550, 2018.

[6] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *CVPR*, pp. 86–94, 2017.

[7] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *CoRR*, vol. abs/1706.06083, 2018.

[9] Z. He, W. Wang, J. Dong, and T. Tan, "Transferable sparse adversarial attack," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14963–14972, June 2022.

[10] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *ICLR*, 2019.

[11] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *CoRR*, vol. abs/1811.12231, 2018.

[12] H. Wang, Z. He, Z. C. Lipton, and E. P. Xing, "Learning robust representations by projecting superficial statistics out.," in *ICLR*, OpenReview.net, 2019.

[13] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The pitfalls of simplicity bias in neural networks," vol. 33, pp. 9573–9585, 2020.

[14] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, pp. 665–673, nov 2020.

[15] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch gaussian augmentation," *CoRR*, vol. abs/1906.02611, 2019.

[16] I. Vasiljevic, A. Chakrabarti, and G. Shakhnarovich, "Examining the impact of blur on recognition by convolutional networks," *CoRR*, vol. abs/1611.05760, 2016.

[17] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

[18] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *CoRR*, vol. abs/1805.09501, 2018.

[19] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," *CoRR*, vol. abs/1905.00397, 2019.

[20] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "AugMix: A simple data processing method to improve robustness and uncertainty," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[21] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer, "The many faces of robustness: A critical analysis of out-of-distribution generalization," *ICCV*, 2021.

[22] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," *CoRR*, vol. abs/1906.08988, 2019.

[23] J. A. Hirsch, J.-M. Alonso, R. C. Reid, and L. M. Martinez, "Synaptic integration in striate cortical simple cells," *Journal of Neuroscience*, vol. 18, no. 22, pp. 9517–9528, 1998.

[24] T. Z. Lauritzen and K. D. Miller, "Different roles for simple-cell and complex-cell inhibition in v1," *Journal of Neuroscience*, vol. 23, no. 32, pp. 10201–10213, 2003.

[25] J. S. Anderson, M. Carandini, and D. Ferster, "Orientation tuning of input conductance, excitation, and inhibition in cat primary visual cortex," *J. NEUROPHYSIOL*, vol. 84, pp. 909–926, 2000.

[26] G. Azzopardi, A. Rodríguez-Sánchez, J. Piater, and N. Petkov, "A push-pull corf model of a simple cell with antiphase inhibition improves snr and contour detection," *PLoS ONE*, vol. 9, p. e98424, 07 2014.

[27] D. Melotti, K. Heimbach, A. Rodríguez-Sánchez, N. Strisciuglio, and G. Azzopardi, "A robust contour detection operator with combined push-pull inhibition and surround suppression," *Information Sciences*, vol. 524, pp. 229–240, 2020.

[28] N. Strisciuglio, G. Azzopardi, and N. Petkov, "Robust inhibition-augmented operator for delineation of curvilinear structures," *IEEE Trans Image Process*, vol. 28, no. 12, pp. 5852–5866, 2019.

[29] N. Strisciuglio, M. Lopez-Antequera, and N. Petkov, "Enhanced robustness of convolutional networks with a push–pull inhibition layer," *Neural Computing and Applications*, 2020.

[30] Y.-t. Li, W.-p. Ma, L.-y. Li, L. A. Ibrahim, S.-z. Wang, and H. W. Tao, "Broadening of inhibitory tuning underlies contrast-dependent sharpening of orientation selectivity in mouse visual cortex," *Journal of Neuroscience*, vol. 32, no. 46, pp. 16466–16477, 2012.

[31] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *AISTATS*, vol. 38, pp. 562–570, 2015.

A Push-pull residual and dense layers

In Figure 3, 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 operators.

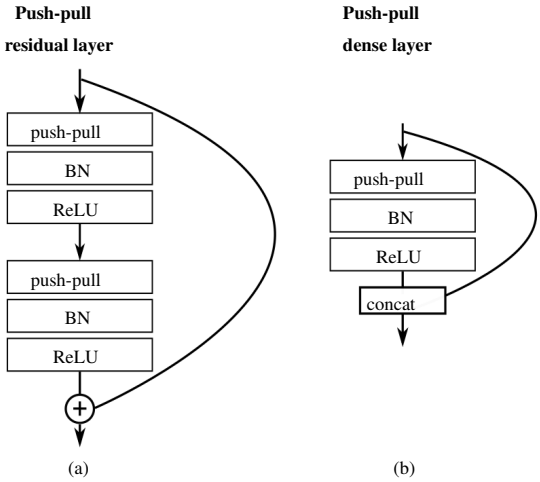

Figure 3: Modified residual and dense layers. In the (a) push-pull residual layer and (b) push-pull dense layer we replace the convolutions with our push-pull operators.

B Training details

For (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.

C Results per corruption type

In Table 3, we report the detailed results achieved on the corruption sets in the CIFAR10-C data set by ResNet networks with and without push-pull inhibition. The push-pull filters are very effective on high-frequency corruptions.

In Table 4, 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 (17) and AutoAugment (18) techniques. The use of the push-pull filters can be combined with data augmentation to further improve the robustness of existing models.

Table 3: Detailed results of ResNet models with push-pull layers on CIFAR10-C per corruption type.

		noise			blur			
	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-pp	8.29	49.90	39.98	37.20	22.30	58.56	29.50	29.68
ResNet-20-b1	8.32	47.60	37.31	38.05	22.31	51.64	29.76	28.95
ResNet-56	6.64	56.72	44.02	44.68	20.02	59.22	26.54	26.73
ResNet-56-pp	7.01	44.92	35.74	33.14	20.88	50.46	27.67	26.58
ResNet-56-b1	7.76	40.17	31.88	29.56	18.56	45.53	25.36	23.36

	weather				digital			
	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-pp	25.59	31.29	17.58	10.34	30.50	20.65	32.70	22.16
ResNet-20-b1	24.00	26.92	18.04	10.29	31.15	19.56	27.95	20.35
ResNet-56	21.99	29.08	14.55	8.30	24.89	18.52	29.79	22.88
ResNet-56-pp	22.32	27.28	15.93	8.68	26.94	18.57	29.06	19.82
ResNet-56-b1	22.51	25.49	16.31	9.78	29.30	17.95	25.94	17.73

Table 4: Results on the corruptions in CIFAR10-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

D Results per perturbation type

We report in Table 5 the detailed performance results, in terms of flip probability, per perturbation type achieved on the CIFAR10-P data set by ResNet models, of different depth, that deploy the push-pull layers. 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.

Table 5: 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**			
	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-pp	0.076	0.097	0.123	0.008	0.039	0.014	0.053	0.055	0.019	0.067
ResNet-20-b1	0.062	0.081	0.114	0.007	0.034	0.014	0.053	0.051	0.018	0.066
ResNet-56	0.0706	0.0943	0.1048	0.0063	0.0333	0.0076	0.0354	0.0458	0.0139	0.0523
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-b1	0.0517	0.0654	0.1007	0.0061	0.0297	0.0130	0.0429	0.0438	0.0158	0.0573

In Table 6 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 (17) and AutoAugment (18) 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 6: Flip probability per perturbation achieved on the CIFAR10-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