Learning the *Essential* in less than 2k additional weights - a single large kernel convolution layer can improve prediction stability under unknown corruptions

Supplementary Material

Overview

In this supplementary material, we provide additional information and details to the experimental results in the main paper. In particular, we provide

- More information and analysis on the employed datasets in section 7.
- More details in the experiment setup in section 8.
- Additional Experimental results including more details on the aggregated results given in the main paper, an ablation on kernel sizes, an ablation on thresholded kernel performance over thresholds and ablation on $L1$ prior regularization strengths in section 9.
- An analysis of the optimized kernels in section 10.
- An extended discussion of our findings in section 11.

7. Datasets

To train the different models, subsets of the ImageNet [46] dataset are used. To have a widespread experiment set, the ImageNette [27] dataset with ten classes and the ImageNet-100 [55] with 100 classes from ImageNet are used.

Furthermore, to evaluate the robustness of these models, the subsets are corrupted via different corruptions from OpticsBench [39] and Common Corruptions [20].

7.1. Clean datasets

ImageNette provides 9,469 training and 3,925 validation images on ten diverse ImageNet classes. We use the ImageNette2 version in all experiments, publicly available from [27]. This allows extensive model training experiments to be carried out with limited computing resources. In addition, some of the experiments were carried out on ImageNet-100, consisting of 100 randomly selected classes from ImageNet-1k with a total of 128k training and 5k validation images. Each class has approximately 1300 training and 50 validation images. As there are no extra test labels, we test on the validation images. The validation images are kept from training. Finally, two experiments are conducted on ImageNet-1k, which consists of 1,000 classes with a total of 1,281k training and 50k validation images.

7.2. Common Corruptions

The diverse common corruptions [20] consist of digital, weather, noise and blur corruptions aimed at benchmarking DNNs in safety-critical applications. We use 16 different algorithmically generated corruptions, organized into five different severities. Besides the 15 benchmark corruptions we also include the saturate corruption from [20] to test for more color corruptions. The individual corruptions are shown in Fig. 8 for severity 4 out of 5. Following [20] images are saved using light JPEG compression, which means that other corruptions are also lightly corrupted by the JPEG compression.

7.3. OpticsBench

The recently published OpticsBench [39] provides additional blur corruptions obtained from optics, which are obtained by convolving images with a given 3D kernel. The blur kernels (x,y,color) are color dependent and have diverse shapes. They are intended as base classes for primary aberrations and complement the blur corruptions from Common Corruptions [20]. Fig. 8 shows the different OpticsBench blur corruptions in the upper left along with the corresponding blur kernel in linear scaling. Coma and astigmatism are shown in one orientation. As the authors use the defocus blur from [20] as a baseline, we also visualize the corresponding defocus blur kernel in Fig. 8.

7.4. Grouping Corruption Types

Combining the two corrupted datasets, we evaluated all models on 24 different corruptions, with each consisting of five severities. The high number of corruptions provides a broad and stable view on the robustness of our models against such corruptions. However, to visualize the results of our evaluation, we grouped corruptions in five super-categories: noise, blur, compression, weather, and color.

The five super-categories with the corresponding corruptions can be examined in Fig. 8. As the first super-category, we combined all noise generating corruptions from the Common Corruptions into *noise*. Furthermore, we grouped all blur corruptions from the Common Corruptions [20] with all corruptions from OpticsBench [39] into the super-category *blur*, as they all mimic different blur types. Similar to [20] we combine all weather corruptions into the super-category *weather*. However, we shifted the brightness corruption to the super-category *color*, as brightness, contrast, and saturate behave similar. The last super-category combines three different image transformation corruptions into the super-category *compression*.

Each of the evaluated corruptions and super-categories have five severities. In addition to the spatial domain visualization in Fig. 8, the super-categories are visualized in the frequency domain, in Fig. 9 to 13. Each of these figures is divided into six columns, in which the first five

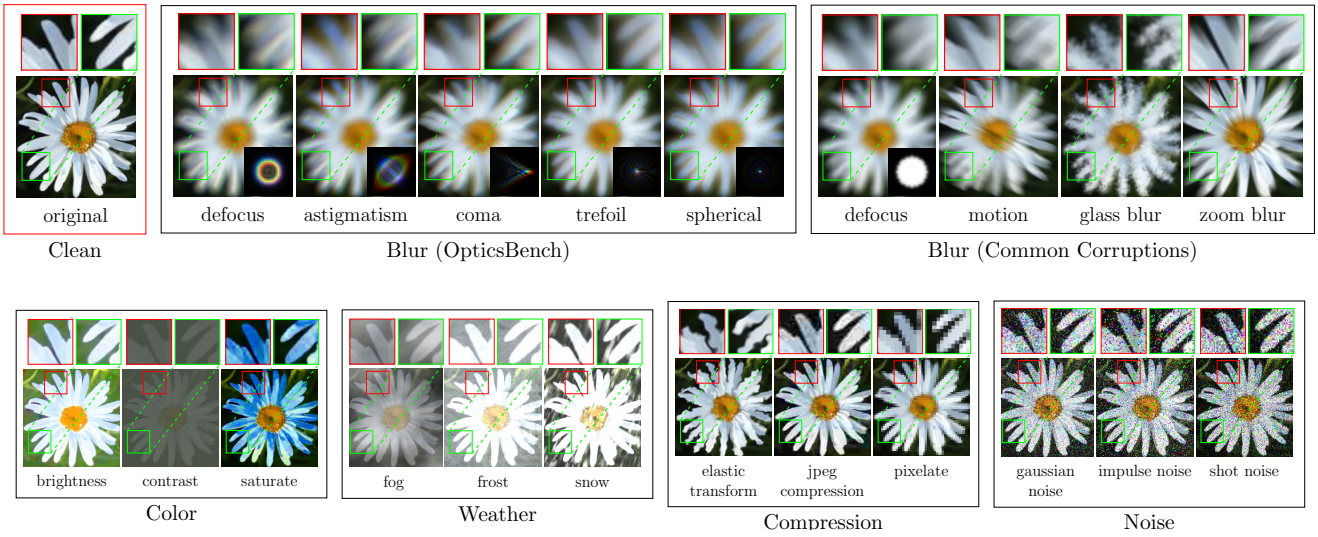


Figure 8. Overview of used corruptions applied to an ImageNet sample. The first row shows the unmodified image (upper left) and various blur corruptions from [39] (left box) and [20] (right box). The second row shows the remaining common corruptions from [20] grouped to color, weather, compression and noise (from left to right). All shown corrupted images represent severity 4 out of 5.

columns represents one of the super-categories. For every super-category, the average Fourier transformation magnitude over 100 images is displayed in the top row. The bottom row corresponds to the absolute difference of corrupted average Fourier transformation magnitude against the clean data average Fourier transformation magnitude. The last column represents the average frequency magnitude of 100 clean data images.

8. Experiment setup

All models are trained on clean data, without any data augmentation. To guarantee a comparability between all models from the same architecture type, we only added our proposed input layer to the corresponding models. Therefore, we did not change any hyperparameters in-between the training of the same architecture type. The hyperparameters for the training are used from [37].

9. Additional Experimental Results

To have a more holistic view on the performance of models with our proposed pre-pend layer, multiple additional experiment are conducted. This section contains multiple detailed evaluation results from models trained on the ImageNette [27] dataset. The next subsection (Sec. 9.1) indicates the reasoning for the proposed kernel size. Section 9.2 presents more experiments conducted with a static filter (class II). In Sec. 9.3 and 9.4, more experiments can be examined on class III kernels. More ImageNet-100 training results are introduced in Sec. 9.5. Additional to the corruption robustness, Sec. 9.6 indicates a higher robustness against adversarial attack by adding our proposed trainable pre-processing layer to existing models.

In Fig. 14 the same pattern, as in Fig. 1, can be inspected. The convolutional preprocessing (ours) increases the performance of the AlexNet model on all corruptions and severities of the OpticsBench [39] dataset.

An increase of accuracy over the most corruptions from Common Corruptions [20] and OpticsBench [39], especially on higher severities are displayed in Fig. 16 to 22.

9.1. Kernel Sizes

To explore a suitable kernel size for our proposed layer, we conducted six experiments with slightly increasing kernel sizes. The results over the clean data, the Common Corruption corrupted data, and the OpticsBench corrupted data, is displayed in Table 5. On the clean data, the baseline, without any prior layer to the mode, outperforms all kernel sizes. Furthermore, the models with a prior trainable layer and a small kernel size (7×7 and 15×15) achieve a higher accuracy on the clean data, than models with a large kernel size, such as 31×31 and 35×35. However, on the corrupted datasets (OpticsBench and Common Corruptions), the models with a larger kernel size outperform models with smaller kernel sizes and the baseline significantly. The best performance on both corrupted datasets and a just slightly worse performance on the clean dataset is achieved by the model

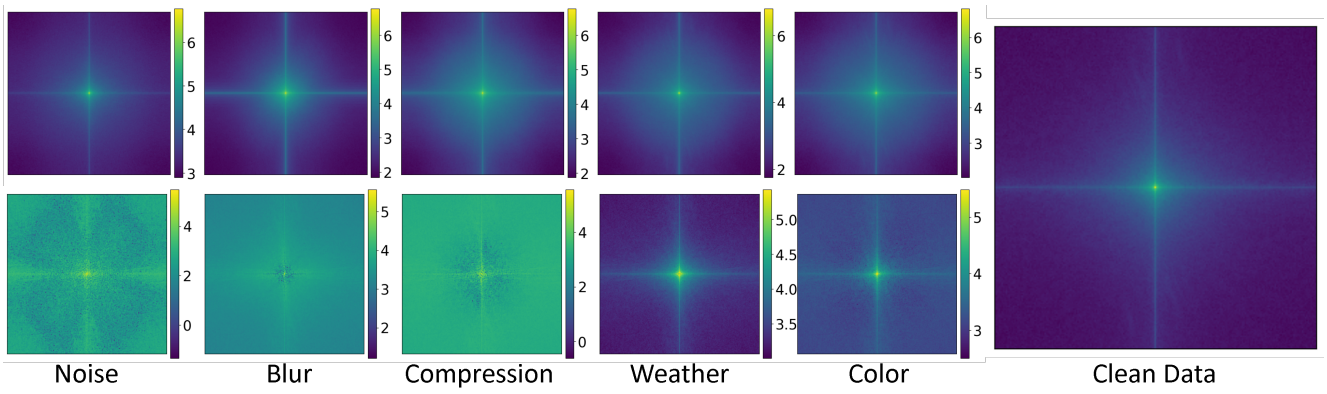


Figure 9. Corruption Severity 1 - Average Fourier transformation magnitude over 100 ImageNet images for each super-category (column 1–5) and the clean data (column 6). Top: The average Fourier transformation magnitude. Bottom: The difference between the corrupted average Fourier transformation magnitude and the clean average Fourier transformation magnitude.

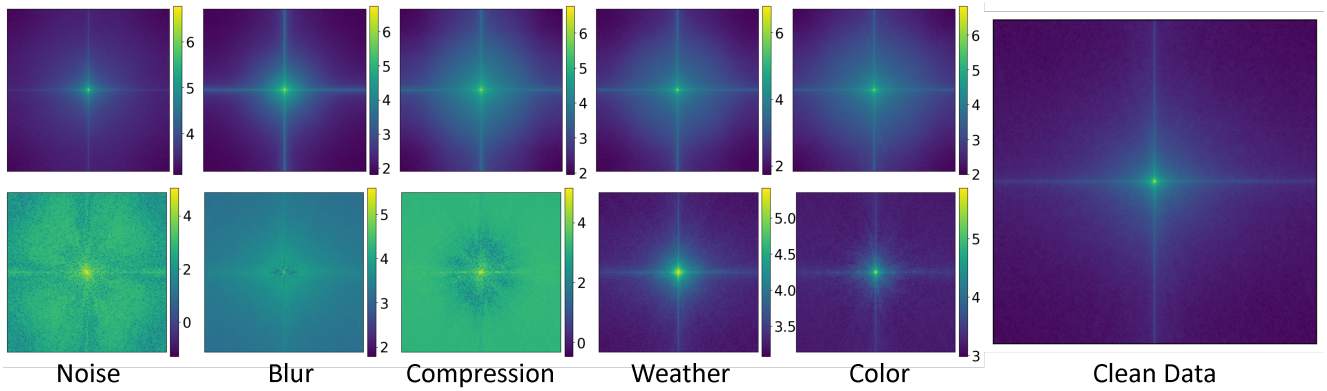


Figure 10. Corruption Severity 2 - Average Fourier transformation magnitude over 100 ImageNet images for each super-category (column 1-5) and the clean data (column 6). Top: The average Fourier transformation magnitude. Bottom: The difference between the corrupted average Fourier transformation magnitude and the clean average Fourier transformation magnitude.

with the kernel size of 25×25. Therefore, we used this kernel size for all other experiments, if not stated differently.

9.2. Static Kernels (Class II)

To investigate the extent to which kernel training improves prediction stability, we freeze different convolutional layer initializations. These static filters are then compared to both the baseline and the fully trainable filters. We investigate two blur filters, one rotational-symmetric Gauss blur filter and a directional (horizontal coma) blur filter obtained from OpticsBench [39]. These have both lowpass characteristic and remove high frequency content. The color distortion filter aims to split color-specific information by translation. The results on clean data and the different corruption benchmarks on ImageNette are shown in Table 6. It is noticeable that the two blur filters have an in-domain accuracy comparable to the fully trainable or the baseline model, while the

Kernel size	CD	OB [39]	CC [20]
Base	***0.800**	*0.592	*0.487
7	0.782	0.616	0.526
15	0.790	0.575	0.532
21	0.765	0.610	0.530
25	*0.775	***0.685**	***0.565**
31	0.707	0.681	0.556
35	0.713	0.678	0.549

Table 5. Results on ImageNette for ResNet50 and different kernel size for the proposed trainable convolutional preprocessing layer. CD = Clean Data, OB = OpticsBench, CC = Common corruptions. * = average from multiple seeds. The results on the two corruption benchmarks are averaged across severity and corruption

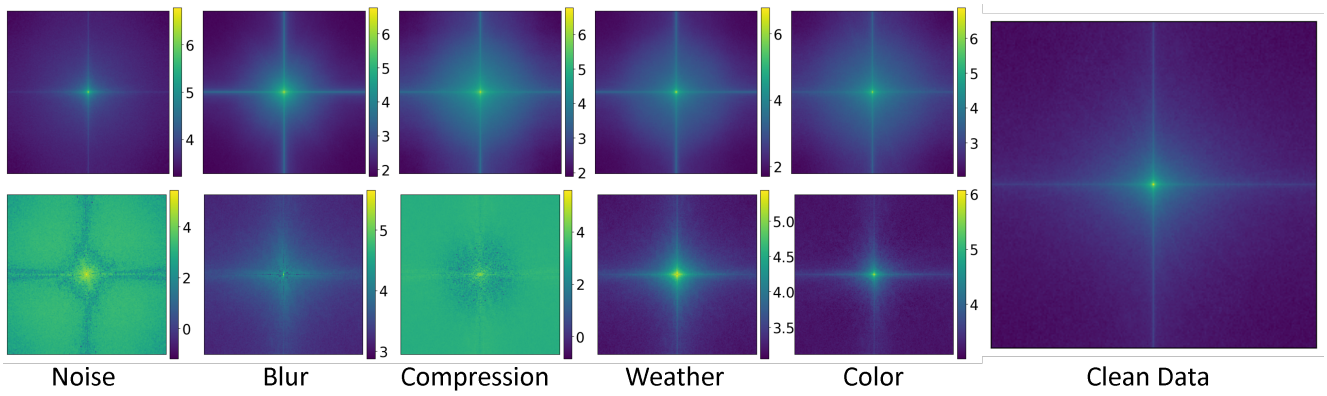


Figure 11. Corruption Severity 3 - Average Fourier transformation magnitude over 100 ImageNet images for each super-category (column 1-5) and the clean data (column 6). Top: The average Fourier transformation magnitude. Bottom: The difference between the corrupted average Fourier transformation magnitude and the clean average Fourier transformation magnitude.

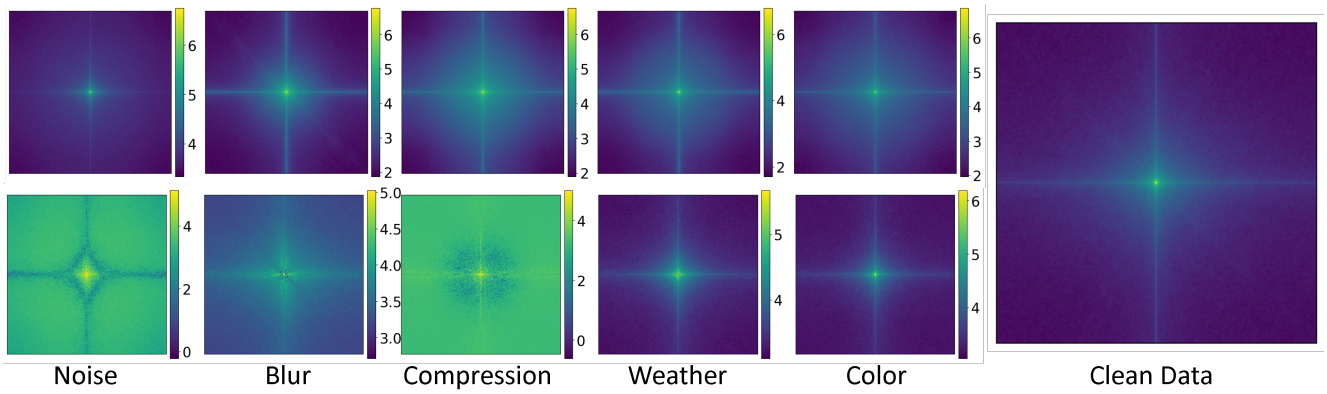


Figure 12. Corruption Severity 4 - Average Fourier transformation magnitude over 100 ImageNet images for each super-category (column 1-5) and the clean data (column 6). Top: The average Fourier transformation magnitude. Bottom: The difference between the corrupted average Fourier transformation magnitude and the clean average Fourier transformation magnitude.

color distortion kernel model is significantly lower. Overall, only the Gaussian blur and color distortion models perform better than the baseline on the two corruption benchmarks. The directional blur model performs worst on corruptions, but second best on clean data.

9.3. Detailed plots of Thresholded Kernel Performance

Fig. 7 shows, the nearly constant performance on the OpticsBench dataset, while reducing the frequencies, with small coefficients, in the trainable kernel. To show this effect in a little more detail, Fig. 23 presents the evaluation results of only one training run. To illustrate the model performance of class III kernels, we trained on MobileNet v3 large model with our proposed convolutional preprocessing layer and no restrictions, such as $L1$ prior. Subsequent to the training, we evaluated the model multiple times with different shares of removed frequencies of the proposed kernel

Kernel type	class	CD	OB [39]	CC [20]
None (Base)	-	***0.800**	*0.592	*0.487
Preserve content	I	0.754	0.476	0.513
Preserve content large	I	0.645	0.478	0.440
Fully trainable	II	*0.775	*0.685	*0.565
L1 prior	III	0.712	**0.699**	**0.567**
Random initialization	II	0.711	0.673	0.550
Con2D KS=25	II	0.655	0.601	0.501
Directional blur filter	II	0.778	0.437	0.345
Gauss blur filter	II	0.764	0.668	0.541
Color distortion	II	0.677	0.660	0.533

Table 6. Results on ImageNette for ResNet50 and different input layer large kernel types. CD = Clean Data, OB = OpticsBench, CC = Common corruptions. * = average from multiple seeds. The results on the two corruption benchmarks are averaged across severity and corruption.

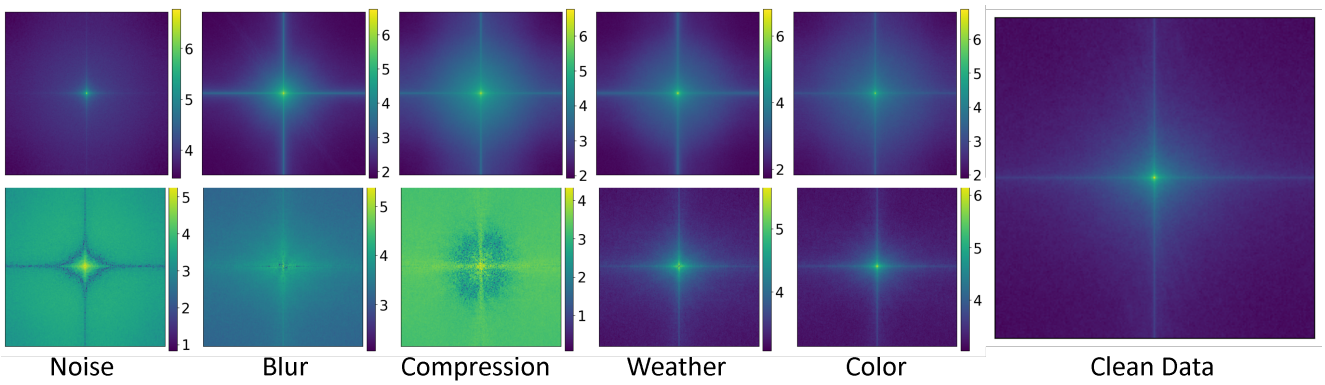


Figure 13. Corruption Severity 5 - Average Fourier transformation magnitude over 100 ImageNet images for each super-category (column 1-5) and the clean data (column 6). Top: The average Fourier transformation magnitude. Bottom: The difference between the corrupted average Fourier transformation magnitude and the clean average Fourier transformation magnitude.

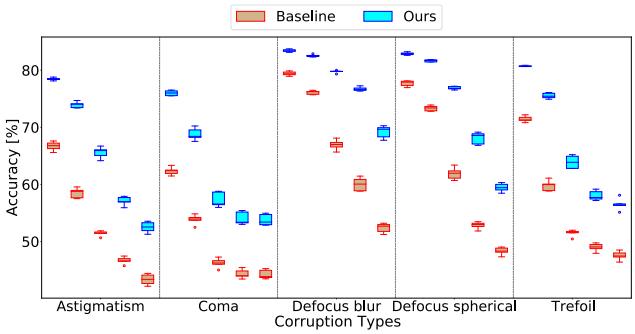


Figure 14. Convolutional preprocessing (ours) can increase the robustness of classification networks against unknown corruptions *without data augmentation*. AlexNet improved with a trainable preprocessing filter evaluated on ImageNette [27] blur and corruptions from OpticsBench [39]. For each corruption type, five levels of severity are shown from left to right. The variation, visualized via the box plots, results from five different seeds per model.

layer. In Fig. 23 the evaluation runs with a lower frequency removal share (0% & 90%) performs better for low severities, but decreases in a faster manner, than evaluation runs with a high frequency removal share (95% & 99%).

9.4. $L1$ prior different regularization strengths

Sec. 9.3 compares already trained class II kernels, which are adjusted to class III kernels, by removing frequencies from the proposed layer kernel. In this section, we further investigate the different class III kernels. Therefore, multiple ResNet50 models with our proposed fully trainable convolutional layer are trained with an $L1$ prior on the frequency coefficients on the trainable kernel of our proposed layer. Each training run uses a different λ to shift the loss more on the frequency coefficients on the trainable kernel. Subsequently, these models are evaluated on the OpticsBench [39] and Common Corruptions [20], which results in Fig. 24. This experiment shows similar results as in Fig. 23, as the results with a lower lambda are decreasing over higher severities faster. Especially, for noise and blur corruptions, this decreasing over severities effect is more significant. However, it seems like models, which are trained and optimized to also minimize the frequency coefficients on the trainable kernel, tend to have a higher overall accuracy for small λs ($\lambda = 0.00001$ to $\lambda = 0.005$), while models with larger λs ($\lambda = 0.001$ to $\lambda = 0.01$), seem to perform more constant over all severities of corruptions.

9.5. ImageNet-100 and ImageNet-1k

Additionally, to all the experiments on ImageNette [27], we also tested the scalability of the proposed convolutional prior layer. Therefore, Fig. 25 to 32 show comparisons of four different models (ResNet, AlexNet, EfficientNet, and MobileNet) on ImageNet-100 [55]. In each figure, we compare a model with our proposed fully trainable convolutional input layer against the baseline. The figures show the performance on the ImageNet-100 corruptions of OpticsBench [39] and Common Corruptions [20]. All the comparisons on ImageNet-100 show, that our proposed model outperforms the baseline on the evaluated corruptions. However, the performance gain is slightly lower than on ImageNette.

While training on the whole ImageNet-1k dataset [46], the performance differences between the baseline and our proposed model is even lower than on ImageNet-100. Only for severity 4 and 5 our model outperforms the baseline. This effect is visualized in Fig. 33 and 34. However, the performance on clean data is also just slightly in favor of the baseline, which is stated in Table 7.

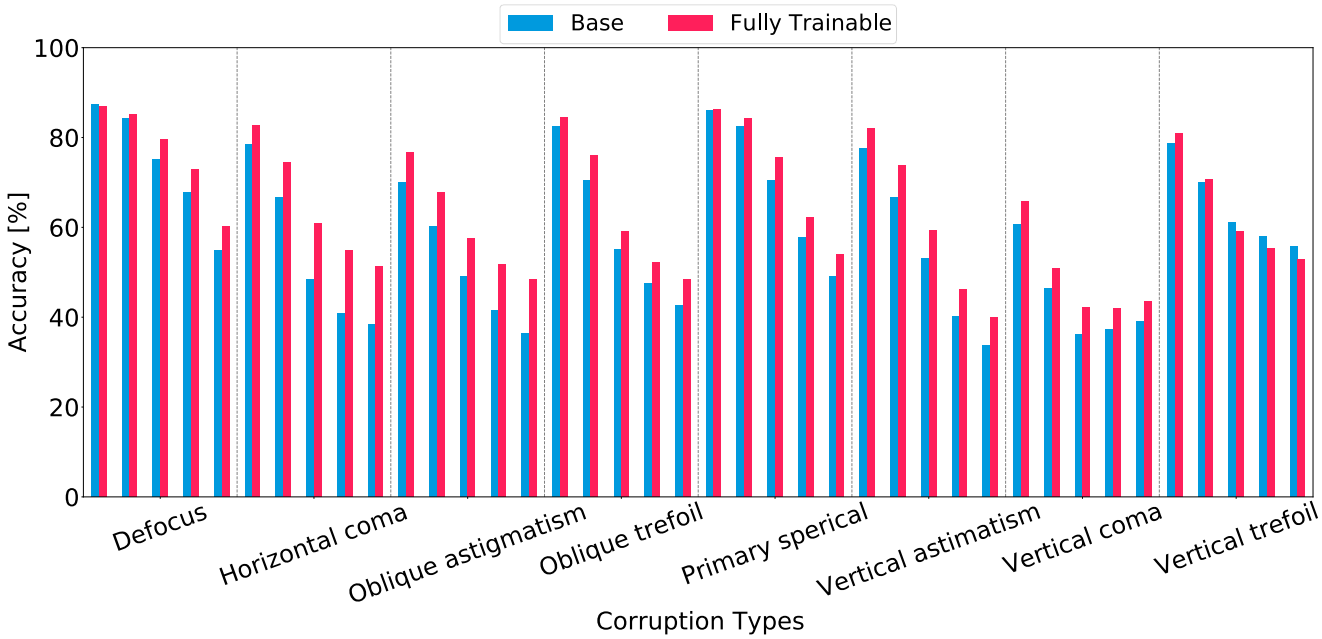


Figure 15. MobileNet v3 large - ImageNette OpticsBench. A comparison of the MobileNet v3 large (Base) and a MobileNet v3 large with our proposed layer (Fully Trainable), on ImageNette OpticsBench.

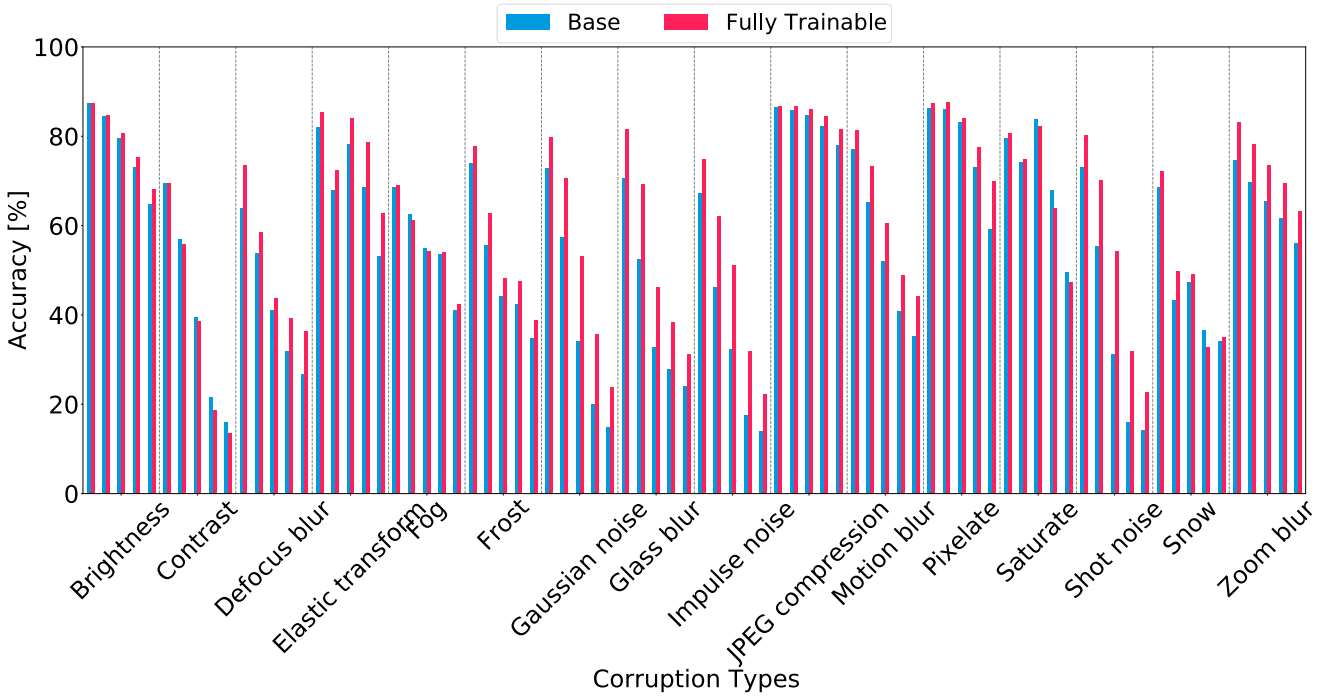


Figure 16. MobileNet v3 large - ImageNette Common Corruptions. A comparison of the MobileNet v3 large (Base) and a MobileNet v3 large with our proposed layer (Fully Trainable), on ImageNette Common Corruptions.

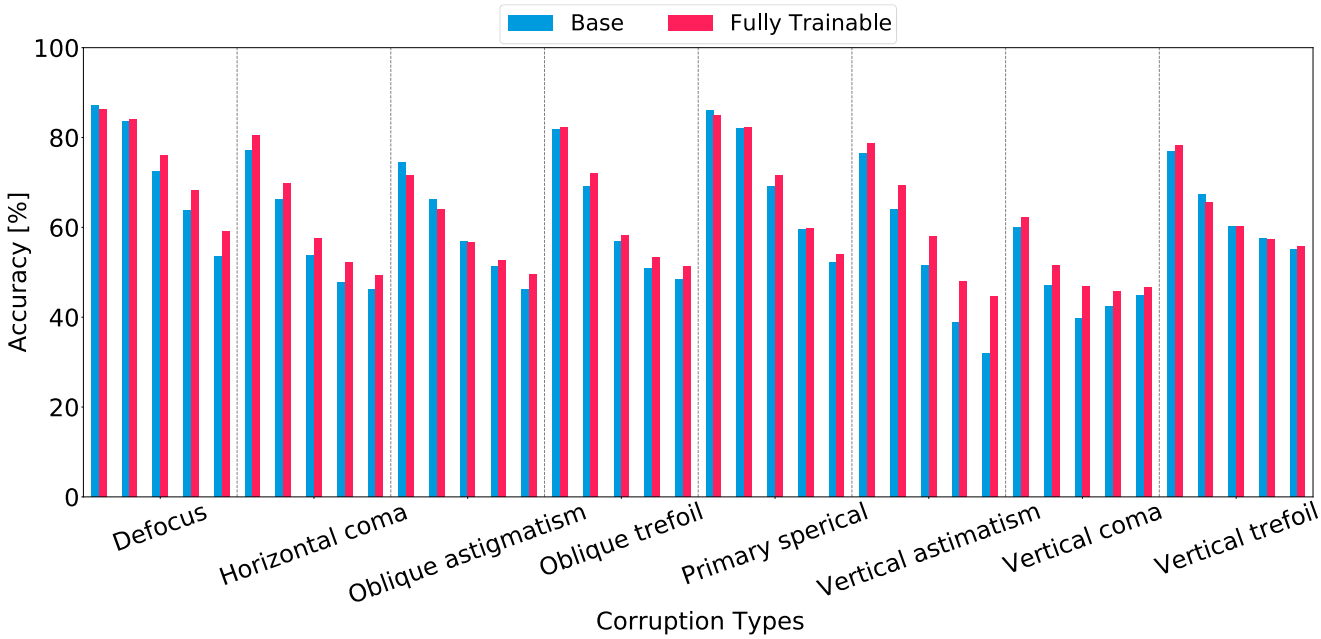


Figure 17. EfficientNet b0 - ImageNette OpticsBench. A comparison of the EfficientNet b0 (Base) and a EfficientNet b0 with our proposed layer (Fully Trainable), on ImageNette OpticsBench.

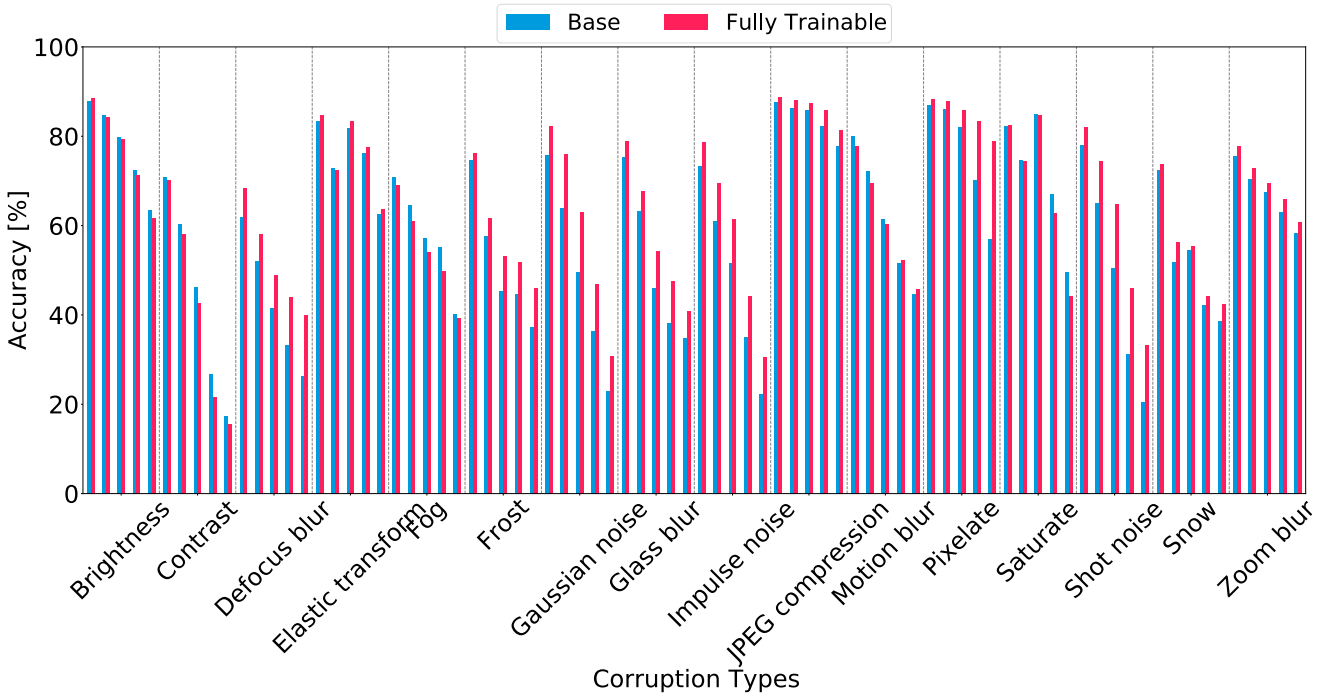


Figure 18. EfficientNet b0 - ImageNette Common Corruptions. A comparison of the EfficientNet b0 (Base) and a EfficientNet b0 with our proposed layer (Fully Trainable), on ImageNette Common Corruptions.

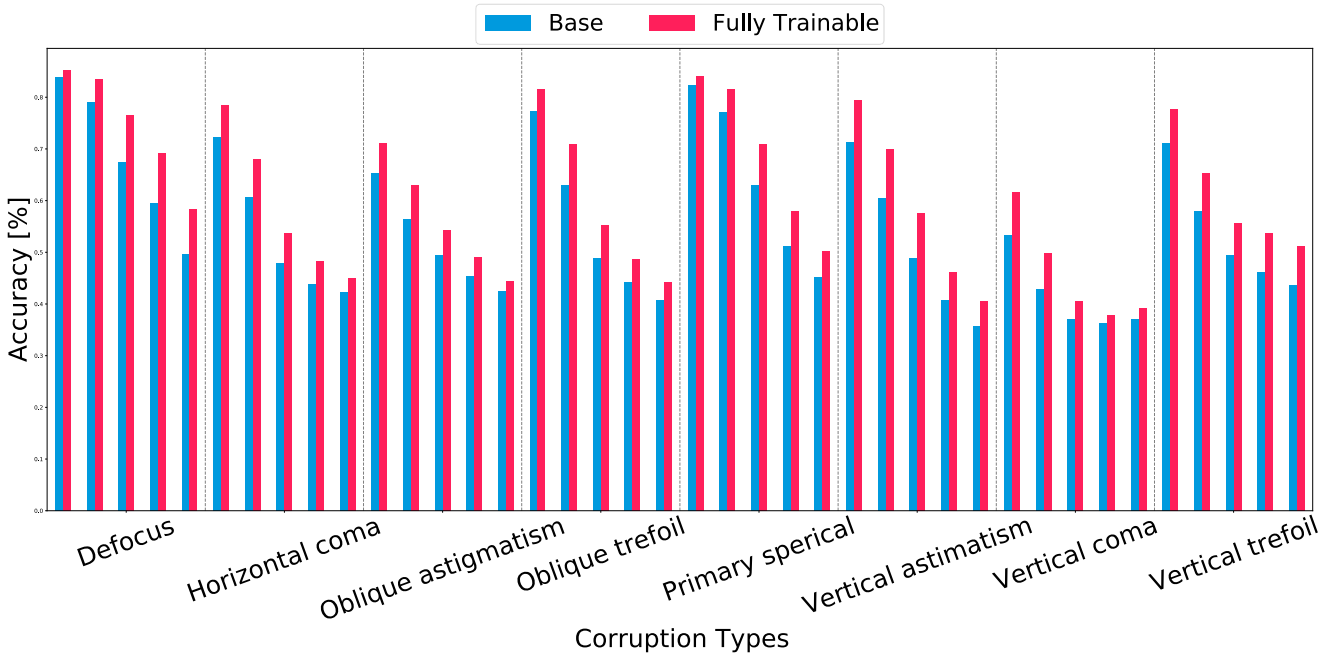


Figure 19. DenseNet161 - ImageNette OpticsBench. A comparison of the DenseNet161 (Base) and a DenseNet161 with our proposed layer (Fully Trainable), on ImageNette OpticsBench.

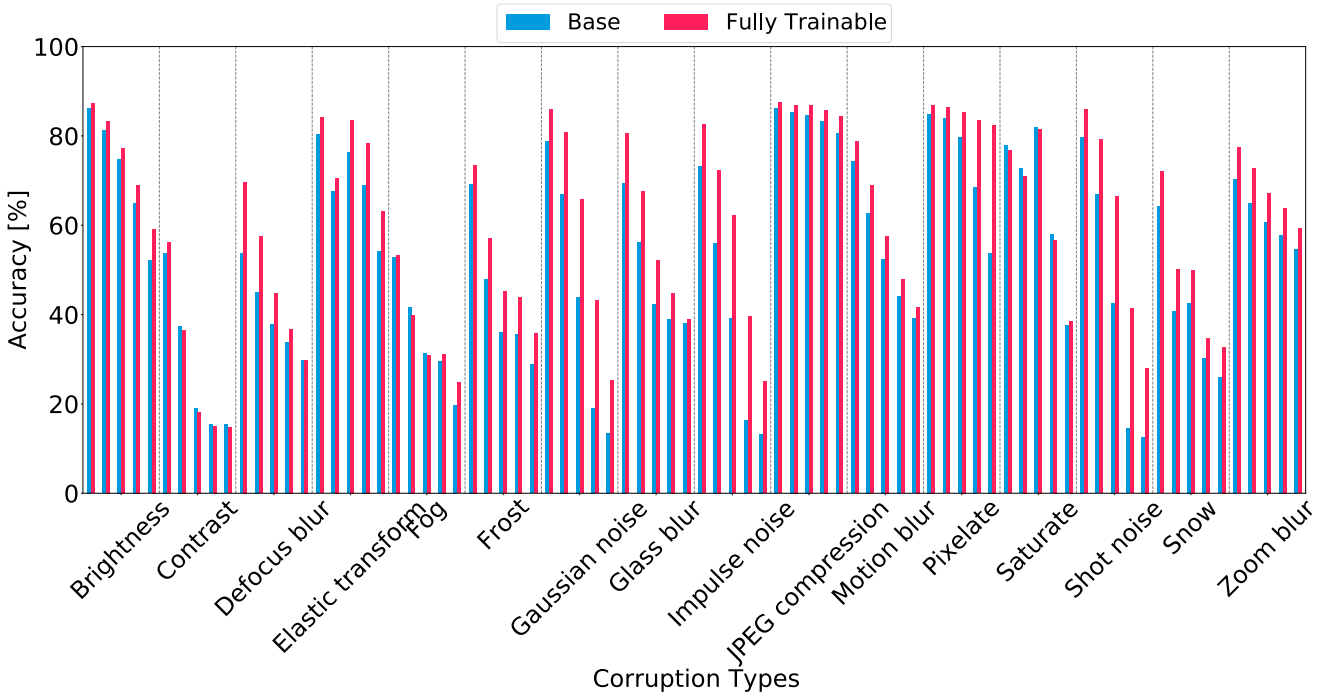


Figure 20. DenseNet161 - ImageNette Common Corruptions. A comparison of the DenseNet161 (Base) and a DenseNet161 with our proposed layer (Fully Trainable), on ImageNette Common Corruptions.

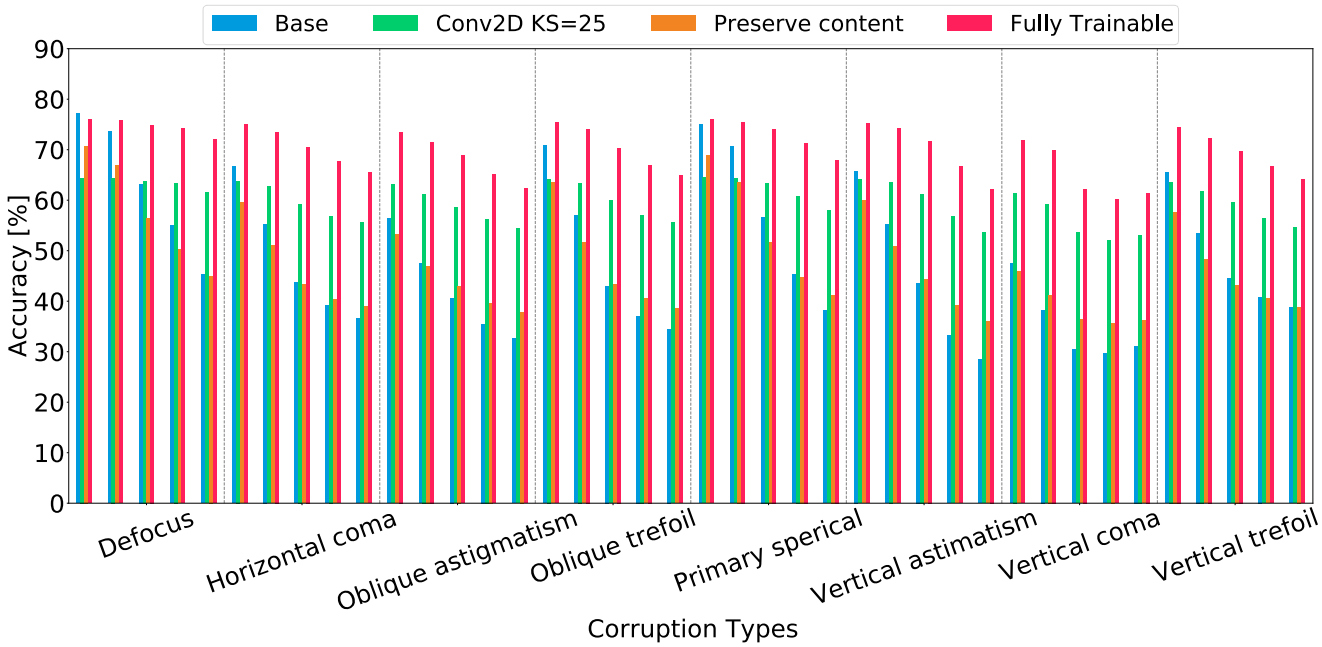


Figure 21. ResNet50 - ImageNette OpticsBench. A comparison of the ResNet50 (Base), the ResNet50 with an additional prior 2D convolutional layer (Conv2D KS=25), a ResNet50 with an additional trainable class I layer (Preserve Content), and a ResNet50 with our proposed layer (Fully Trainable), on ImageNette OpticsBench.

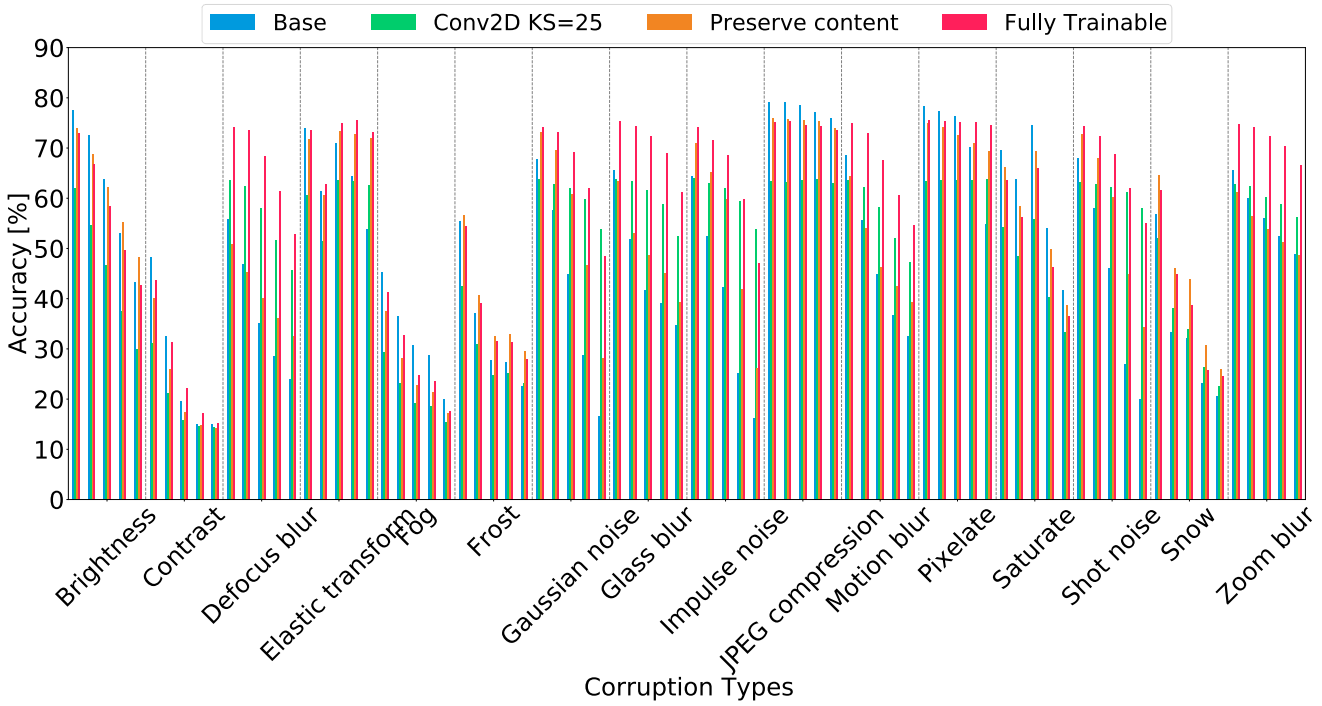


Figure 22. ResNet50 - ImageNette Common Corruptions. A comparison of the ResNet50 (Base), the ResNet50 with an additional prior 2D convolutional layer (Conv2D KS=25), a ResNet50 with an additional trainable class I layer (Preserve Content), and a ResNet50 with our proposed layer (Fully Trainable), on ImageNette Common Corruptions.

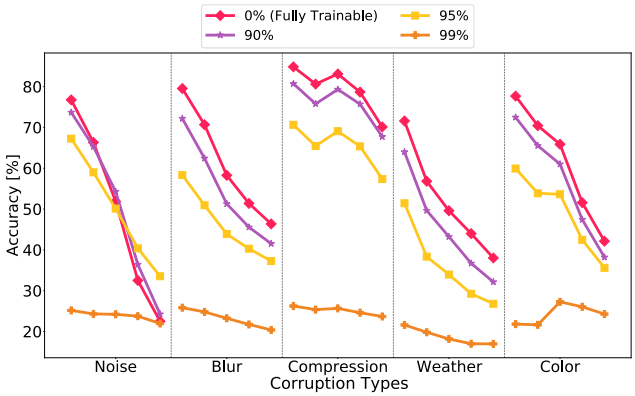


Figure 23. Comparison of frequency thresholded kernels on OpticsBench [39] and Common Corruptions [20]. All evaluation results from this diagram are from the same MobileNet v3 large trainings run, with different thresholding intervals.

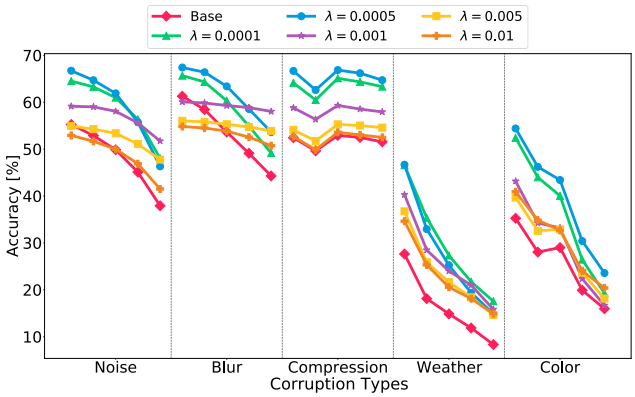


Figure 24. Comparison of models, which are trained with an $L1$ prior. Each model has an ResNet50 architecture, with our additional proposed fully trainable convolutional layer. All these models are trained on ImageNette [27] and evaluated on corruptions from OpticsBench [39] and Common Corruptions [20].

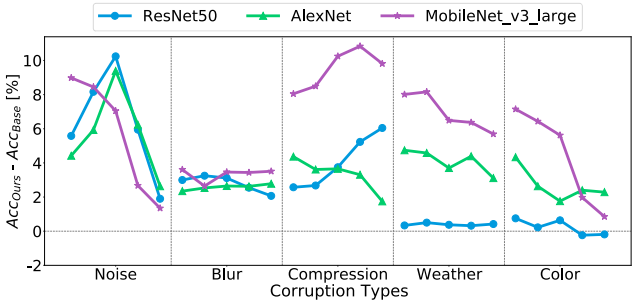


Figure 25. Improvement on ImageNet-100 OpticsBench and Common Corruptions. For each corruption, five levels of severity are shown from left to right. For readability, we summarize different corruption types and take the average.

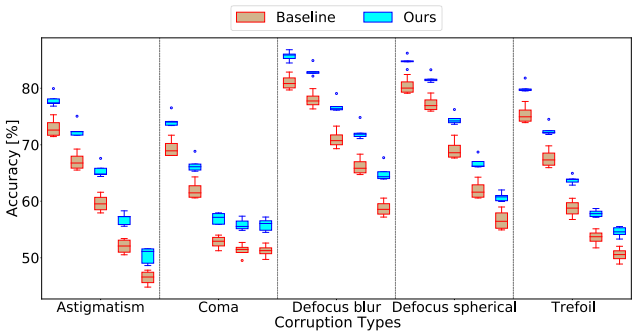


Figure 26. Convolutional preprocessing (ours) can increase the robustness of classification networks against unknown corruptions *without data augmentation*. ResNet50 improved with a trainable preprocessing filter evaluated on ImageNet-100 [55] blur and corruptions from OpticsBench [39]. For each corruption type, five levels of severity are shown from left to right. The variation, visualized via the box plots, results from five different seeds per model.

Model	Version	CD	OB	CC
ResNet50	Base	**0.522**	0.221	0.195
ResNet50	Fully trainable	0.503	**0.226**	**0.197**
MobileNet	Base	**0.646**	0.292	**0.240**
MobileNet	Fully trainable	0.639	**0.296**	0.233

Table 7. Results on ImageNet-1k for ResNet50 and MobileNet v3 large. CD=Clean Data, OB = OpricsBench, CC=Common Corruptions. The results on the two corruption benchmarks are averaged across severity and corruption.

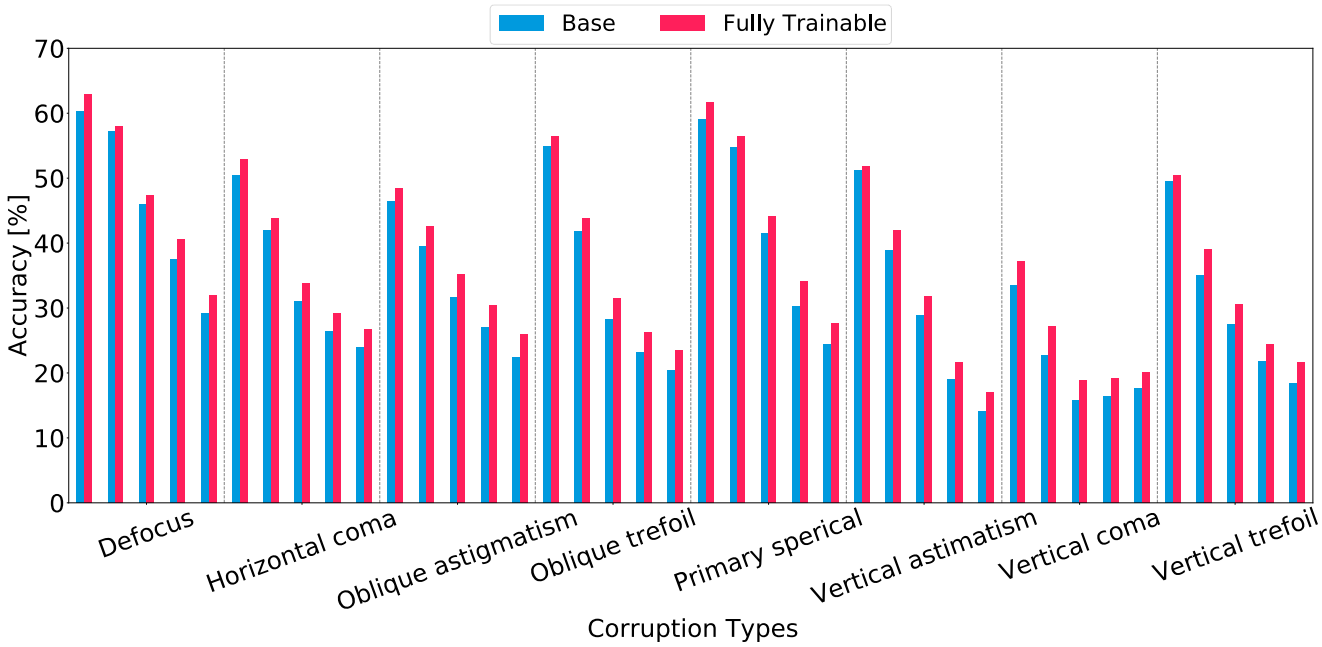


Figure 27. AlexNet - ImageNet-100 OpticsBench. A comparison of the AlexNet (Base) and a AlexNet with our proposed layer (Fully Trainable), on ImageNet-100 OpticsBench.

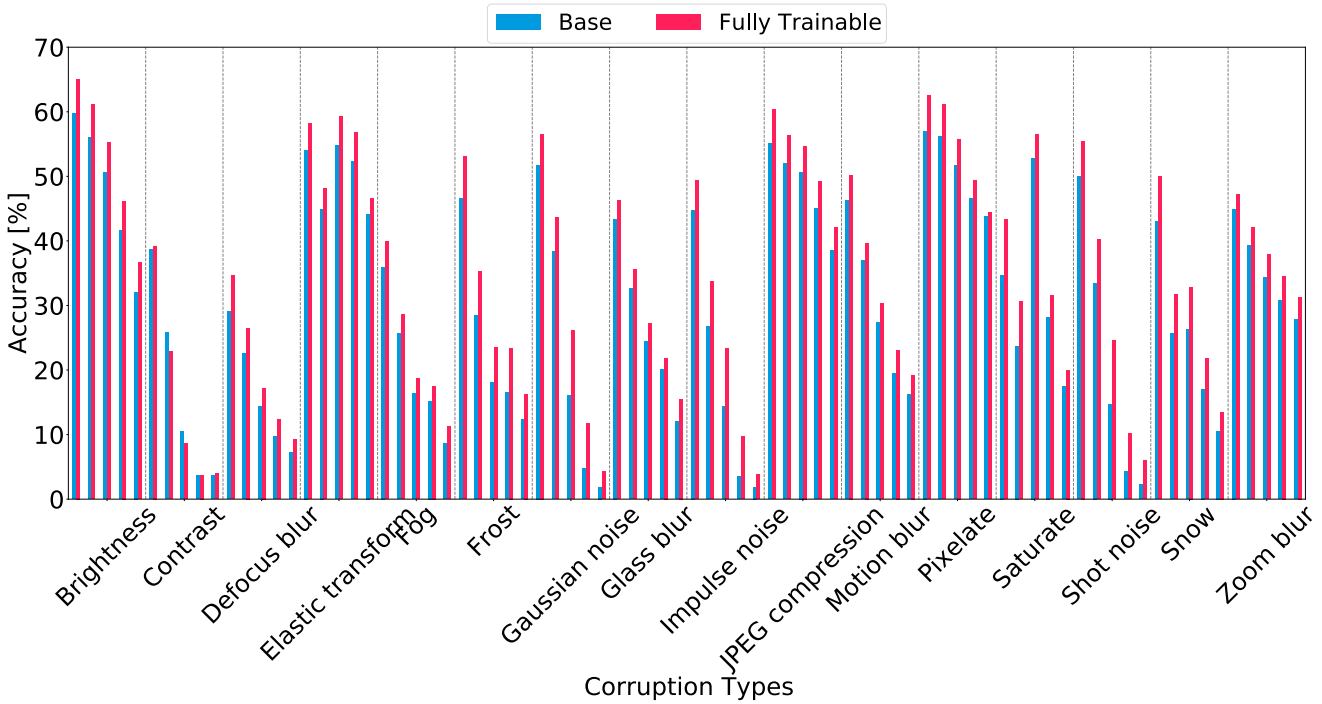


Figure 28. AlexNet - ImageNet-100 Common Corruptions. A comparison of the AlexNet (Base) and a AlexNet with our proposed layer (Fully Trainable), on ImageNet-100 Common Corruptions.

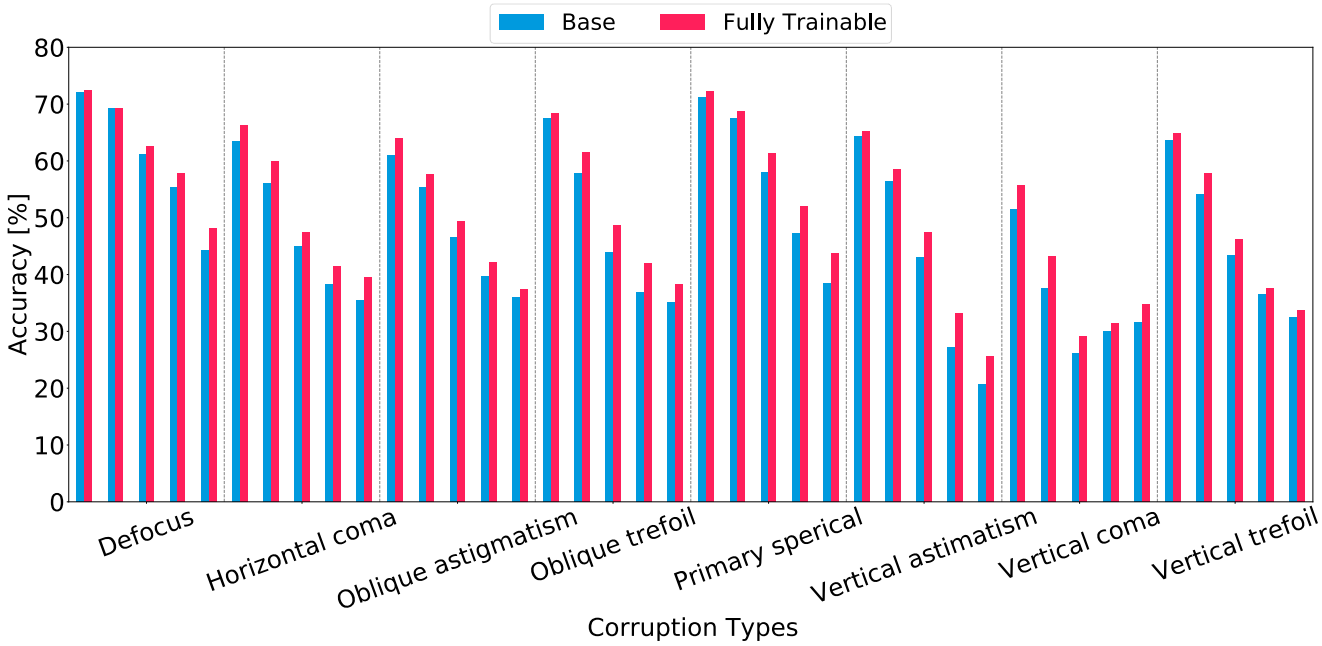


Figure 29. EfficientNet b0 - ImageNet-100 OpticsBench. A comparison of the EfficientNet b0 (Base) and a EfficientNet b0 with our proposed layer (Fully Trainable), on ImageNet-100 OpticsBench.

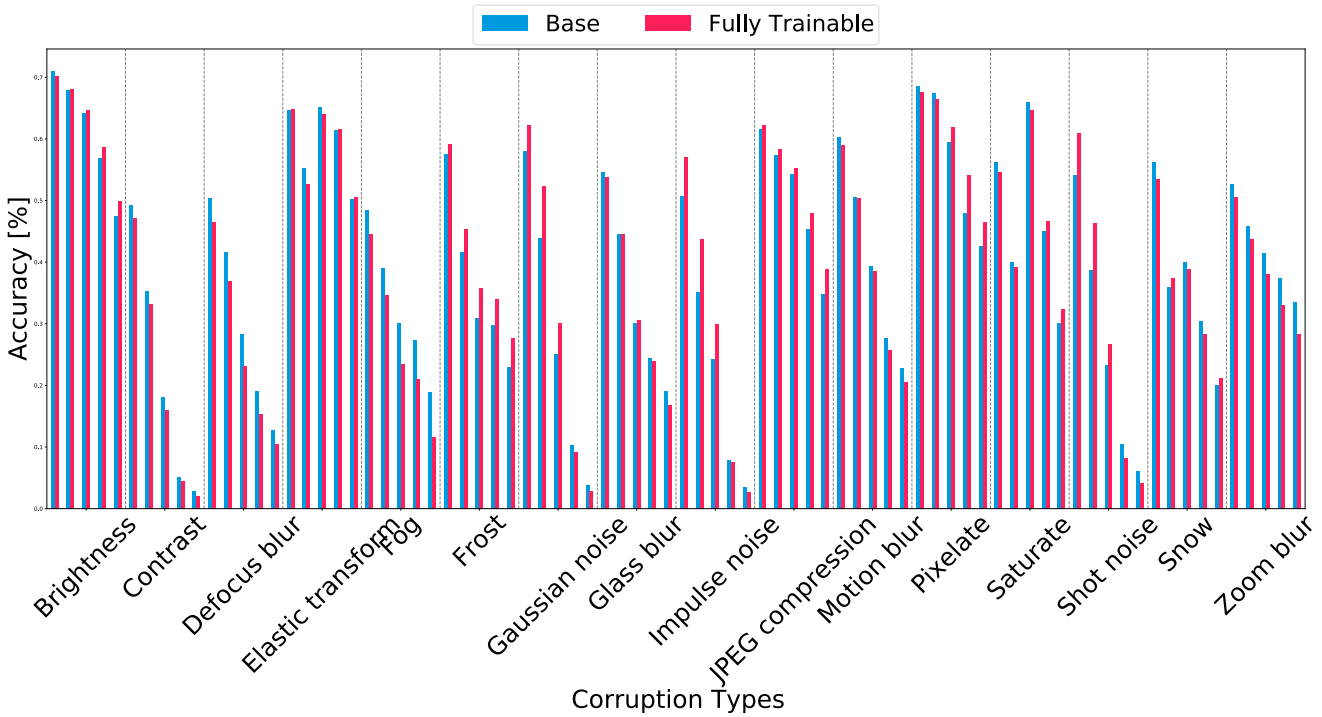


Figure 30. EfficientNet b0 - ImageNet-100 Common Corruptions. A comparison of the EfficientNet b0 (Base) and a EfficientNet b0 with our proposed layer (Fully Trainable), on ImageNet-100 Common Corruptions.

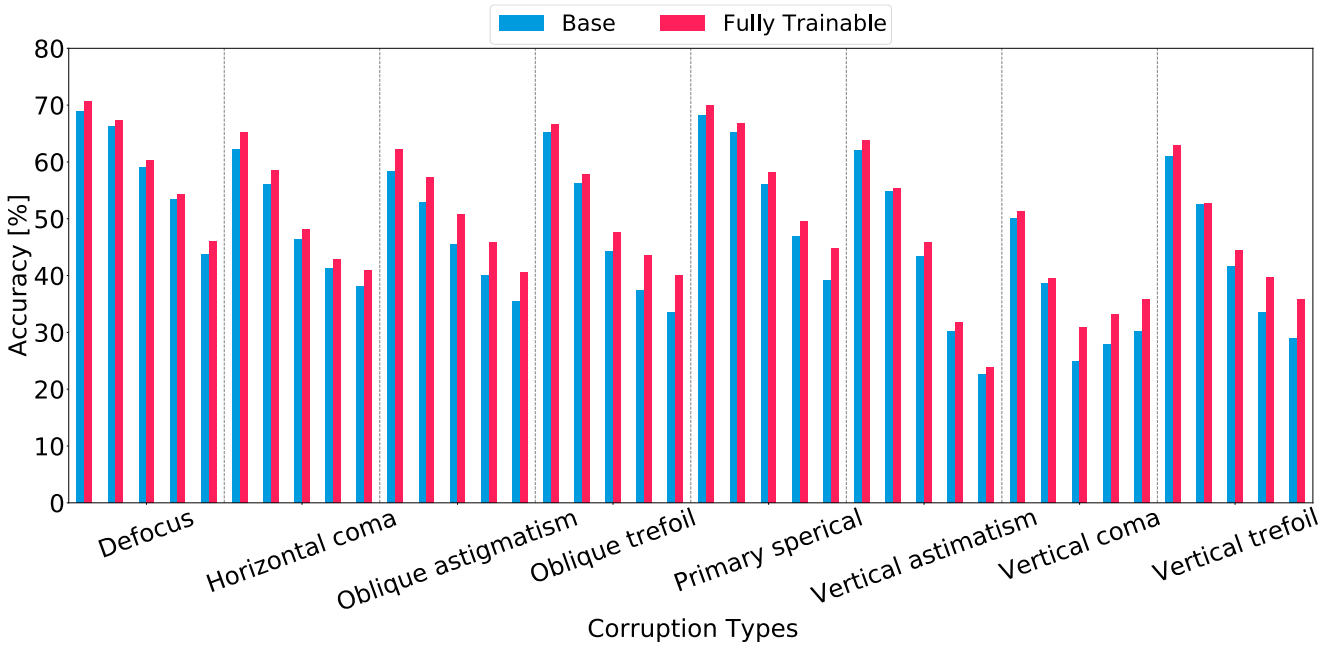


Figure 31. MobileNet v3 large - ImageNet-100 OpticsBench. A comparison of the MobileNet v3 large (Base) and a MobileNet v3 large with our proposed layer (Fully Trainable), on ImageNet-100 OpticsBench.

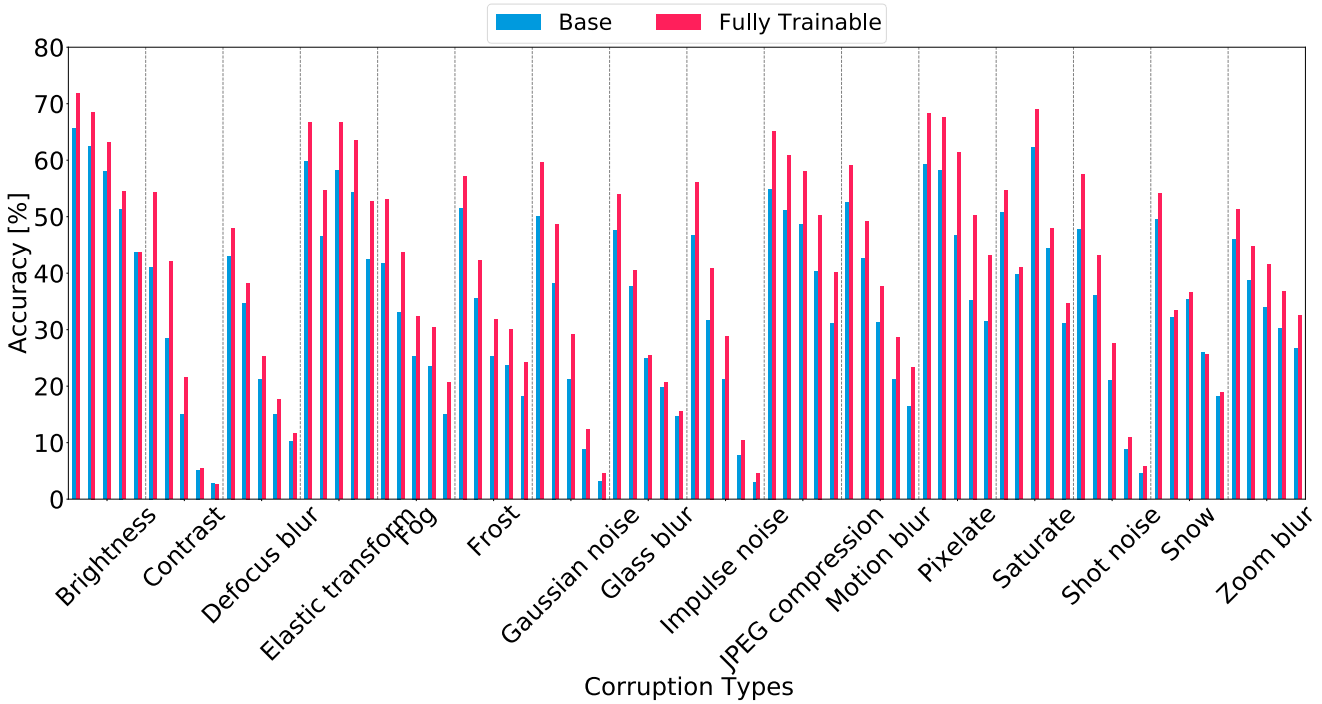


Figure 32. MobileNet v3 large - ImageNet-100 Common Corruptions. A comparison of the MobileNet v3 large (Base) and a MobileNet v3 large with our proposed layer (Fully Trainable), on ImageNet-100 Common Corruptions.

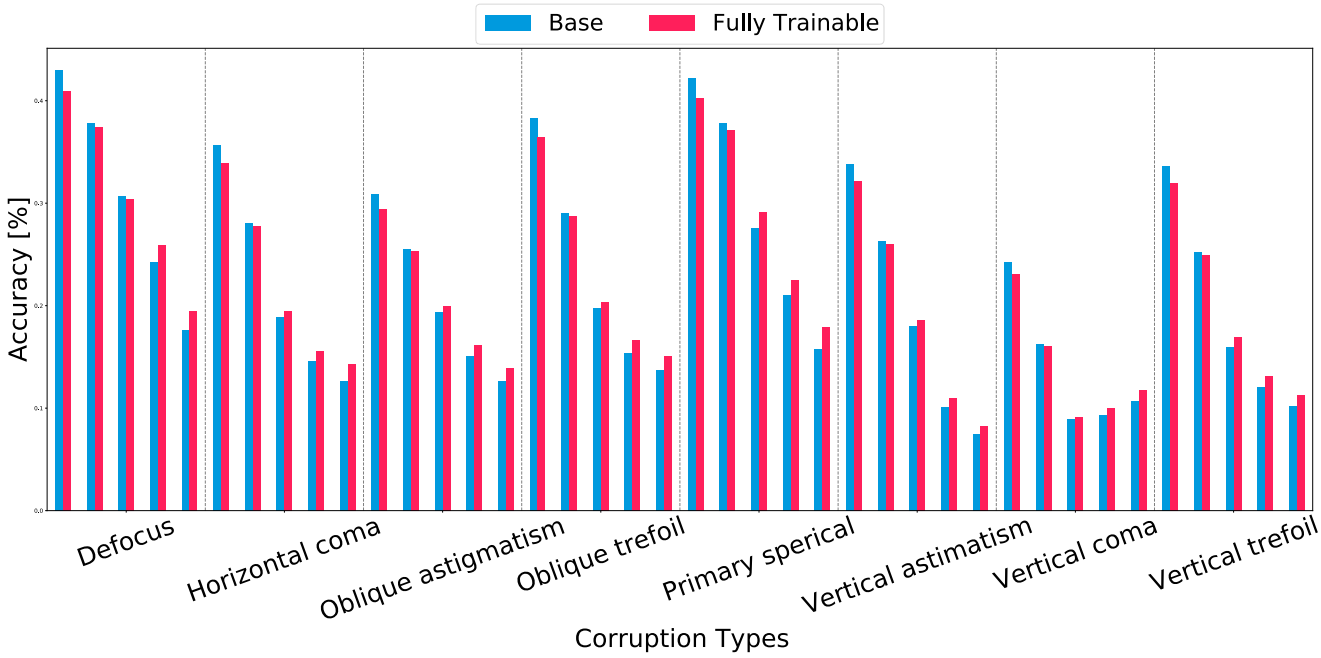


Figure 33. ResNet50 - ImageNet-1k OpticsBench. A comparison of the ResNet50 (Base) and a ResNet50 with our proposed layer (Fully Trainable), on ImageNet-1k OpticsBench.

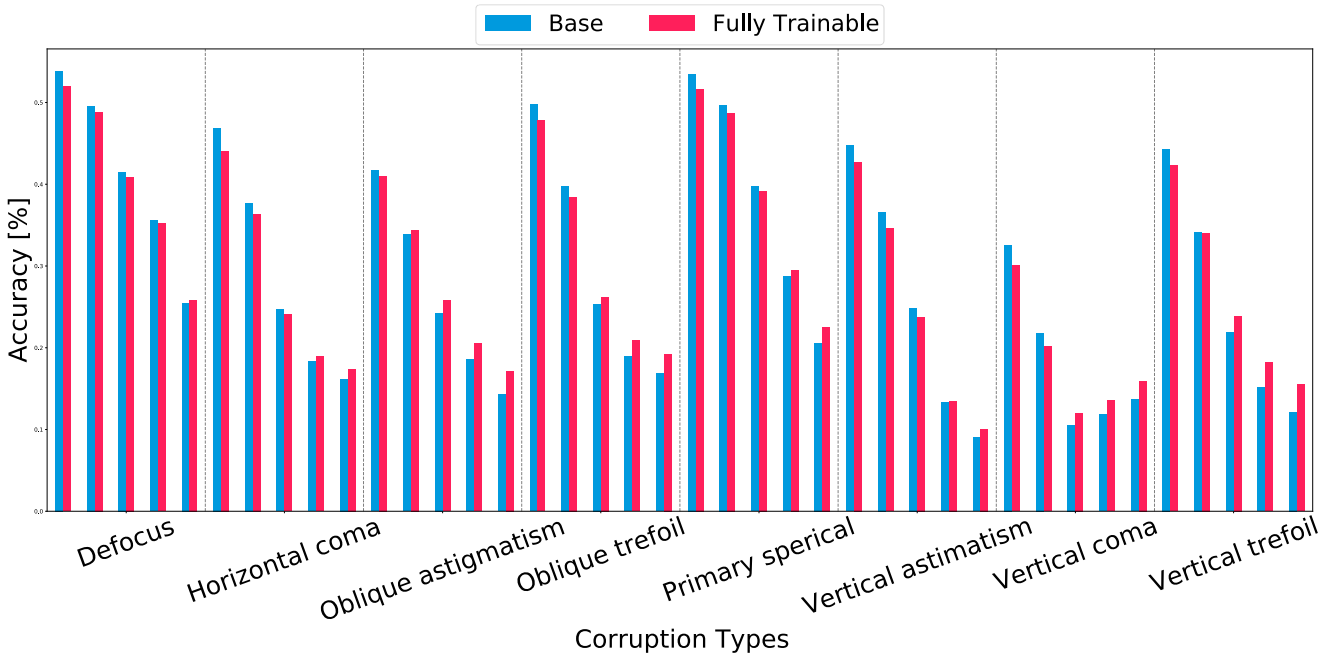


Figure 34. MobileNet v3 large - ImageNet-1k OpticsBench. A comparison of the MobileNet v3 large (Base) and a MobileNet v3 large with our proposed layer (Fully Trainable), on ImageNet-1k OpticsBench.

9.6. Adversarial training and robustness

We further investigate the performance of the proposed input layer on white-box adversarial attacks. Therefore, we trained a ResNet50 with and without our proposed trainable convolutional input layer. The models are trained with the fast gradient sign method (FGSM) [15] and an ϵ of $\frac{2}{255}$ and $\frac{6}{255}$ and evaluated on clean data and under the FGSM adversarial attack, with two different ϵ of $\frac{2}{255}$ and $\frac{6}{255}$.

Model	Version	CD	FGSM $\epsilon = \frac{2}{255}$	FGSM $\epsilon = \frac{6}{255}$
ResNet50	Adv. Baseline	0.665	0.567	0.385
ResNet50	Adv. Trainable	**0.776**	**0.667**	**0.389**
ResNet50	Adv. Baseline	0.534	0.483	0.760
ResNet50	Adv. Trainable	**0.581**	**0.527**	**0.776**

Table 8. Adversarial Results on ImageNette [27] data. CD= Clean Data, FGSM = FGSM attack [15] with two different ϵ of $\frac{2}{255}$ and $\frac{6}{255}$. The two models in the top rows are trained with an ϵ of $\frac{2}{255}$, while the two model on the bottom are trained with an ϵ of $\frac{6}{255}$.

In Table 8 the results of these experiments, indicate the robustness of the models with our preprocessing layer: they are outperforming the baseline on all evaluated data. While both models have similar accuracy under FGSM attacks with ϵ of $\frac{2}{255}$, our model achieves better results on clean data (+11.1% & +4.7%) and with a lower ϵ value of $\frac{2}{255}$ (+10.0% & +4.4%). These increases are possible while only adding one layer with less than 2k of parameters.

10. Analysis of optimized kernels

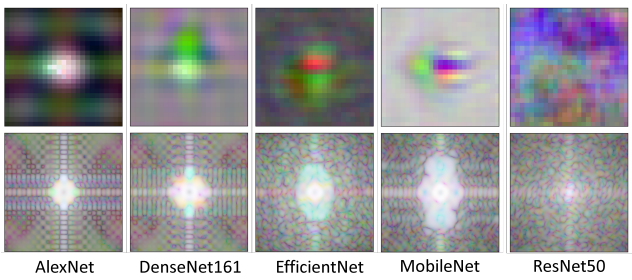


Figure 35. Optimized kernels in the spatial domain (top) and their image-sized frequency magnitudes in $\log_{10}$ scale (bottom) trained on ImageNette.

In this section, we perform an in-depth analysis of the pre-pended kernels. Fig. 35 visualizes optimized class II kernels and their corresponding frequency magnitudes. It can be observed that the optimized kernels for different neural networks are different. In the following, we quantitatively assess the similarities of these kernels, compare the evolution of their spectra, and discuss their condition numbers in finer detail.

10.1. Spectrum correlation

To ensure that our results reflect performance gains due to the proposed method and not because of fortuitous initialization, we experiment with multiple seeds. The kernel is, without change, initialized with a specific coma filter, and the weights of the remaining network are generated pseudo-randomly. We find that the performance of our networks is unaffected. A note-worthy discovery, however, is that despite different initializations, the optimized kernels' spectra are found to be very similar, given the neural network architecture and the input dataset remain the same. Fig. 36 numerically assesses the structural similarity of the spectra of multiple kernels post-optimization. One reason for the observed similarity could be that the proposed kernel learns to map the input to an information-dense subspace most suited to aid the following network in its classification task. Given that the input dataset and the architecture remain invariant, the information-dense subspace should not change as well. Another observation is that given the same input dataset, changing the network changes the mapping learned by the kernel. This suggests that different neural networks predominantly make use of different information content of the unadulterated dataset.

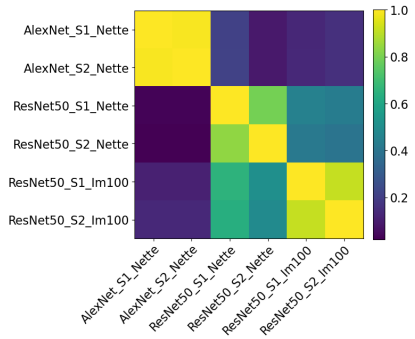


Figure 36. Structural similarity of the magnitudes of the spectra of different kernels. S1 and S2 refer to different seed values, and Im100 and Nette refer to ImageNet-100 and ImageNette, respectively.

10.2. Spectrum evolution

We initialize our kernels with a specific coma filter, which can be observed in Fig. 38 (a) and (d). Moreover, these kernels do not mix channel information. We can observe the evolution of the kernels in the spatial and frequency domains in Fig. 38, when the networks are trained on ImageNet-100. From the figure, it is also apparent that the kernels learn different projections for each color channel.

Fig. 39 compares the evolution of class II and class III kernels. The frequency band axis represents the dimension of a square window, with its center being the center of the frequency-shifted spectrum. Each bar indicates the average

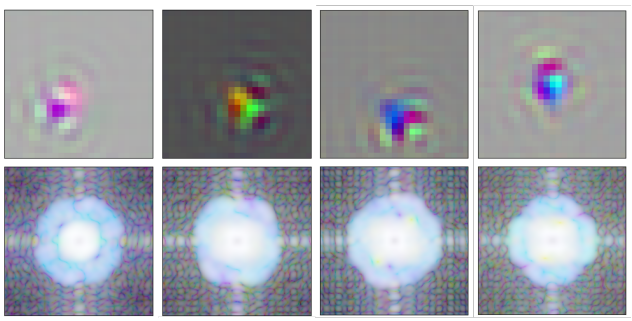
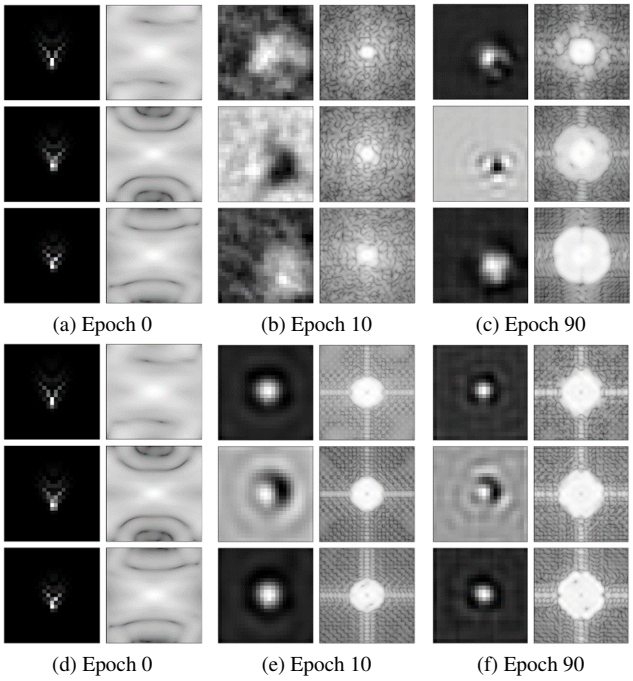


Figure 37. Optimized kernels in the spatial domain (top) and their image-sized frequency magnitudes in $\log_{10}$ scale (bottom) pre-pended to ResNet50 initialized with different seeds and trained on ImageNet-100.

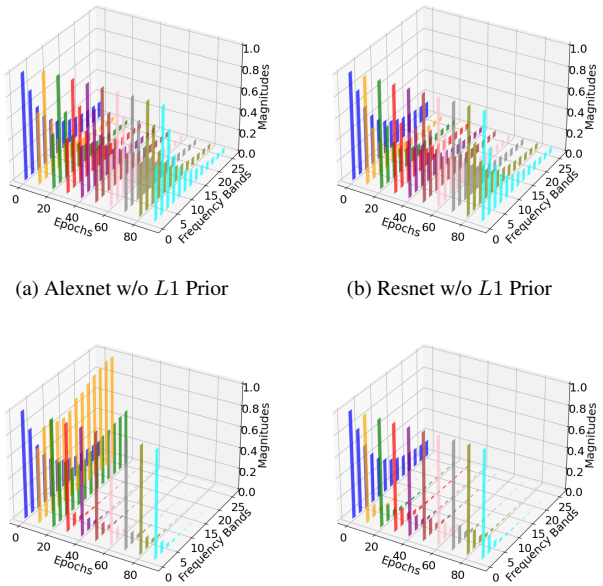


(a) Epoch 0 (b) Epoch 10 (c) Epoch 90

(d) Epoch 0 (e) Epoch 10 (f) Epoch 90

Figure 38. Evolution of kernels in the spatial domain (left) and their frequency magnitudes in $\log_{10}$ scale (right). Kernels in (a-c) and (d-f) were pre-pended to ResNet50 and AlexNet, respectively. Each row corresponds to a color channel.

magnitude of the frequencies in this window, which we refer to as a frequency band. Then, the first bar in each epoch is the magnitude of the DC component of the spectrum and the last is the mean of the magnitudes of the entire spectrum. From Fig. 39, we can observe that the spread of the bars depends on the presence of the $L1$ prior. The $L1$ prior reduces the higher frequency components of the spectrum. Barring the change in prior, the evolution of the spectrum is similar.



(a) Alexnet w/o $L1$ Prior (b) Resnet w/o $L1$ Prior

(c) Alexnet with $L1$ Prior (d) Resnet with $L1$ Prior

Figure 39. Evolution of spectra of (a-b) Fully Trainable and (c-d) $L1$ Prior kernels. The bar height indicates the average of the absolute value of the Fourier coefficients in different frequency bands (DC component in the front). Each epoch is normalized separately.

10.3. Condition numbers

We use the condition number (CN) as one of the two defining characteristics of our classes of kernels. One significant feature of this description is that the convolution kernel, $\cdot * g$, and the corresponding deconvolution kernel, $\cdot * g^{-1}$, share the same CN. This is particularly important for class I kernels because it suggests that the neural network following $\cdot * g$ can perfectly restore the input without being stymied by noise amplification, which distinguishes class I kernels from the rest. In this section, we will derive the equation for CN provided in Sec. 3 and further inspect the CNs of class II kernels.

Derivation of the Condition Number: For an arbitrary invertible matrix A, CN is defined as:

$$CN\left(A\right) = ||A|| \cdot ||A^{-1}|| = ||A^{-1}|| \cdot ||A|| = CN\left(A^{-1}\right),$$

where $||.||$ is the norm of a matrix. The sensitivity of A to noise or the noise amplification possible by the transformation is equal to that of its inverse. Moreover, $||A|| = \sigma_{\max}\left(A\right)$ and $||A^{-1}|| = 1/\sigma_{\min}\left(A\right)$, where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum singular values of the matrix, respectively. In our case, the transformation A is a circulant matrix describing the convolution operation. The eigenvectors of A form an orthonormal Fourier basis. Let

each basis be a column of Q, and let the corresponding eigenvalues λ_i, the Fourier coefficients, form a diagonal matrix, Λ. A can be decomposed as

$$A = Q\Lambda Q^T,$$

as it is a normal matrix. Then using the relation $\sigma_i = \sqrt{\lambda_i\left(A^T A\right)}$,

$$\begin{aligned}\sigma_{\max}\left(A\right) &= \sqrt{\lambda_{\max}\left(A^T A\right)} \\ &= \sqrt{\lambda_{\max}\left(\left(Q\Lambda Q^T\right)^T Q\Lambda Q^T\right)} \\ &= \sqrt{\lambda_{\max}\left(Q\Lambda^2 Q^T\right)} \\ &= |\lambda_{\max}\left(A\right)|.\end{aligned}$$

Similarly, $\sigma_{\min}\left(A\right) = |\lambda_{\min}\left(A\right)|$. Therefore,

$$CN\left(A\right) = \frac{|\lambda_{\max}\left(A\right)|}{|\lambda_{\min}\left(A\right)|} = \frac{1/|\lambda_{\min}\left(A\right)|}{1/|\lambda_{\max}\left(A\right)|} = CN\left(A^{-1}\right).$$

Note, the definition of CN in the main paper does not include $|.|$. This is equivalent to using the magnitudes of the Fourier coefficients instead of the coefficients themselves. However, all the numbers presented are correctly computed.

Class II Kernels: The CN of class I kernels is by their definition equal to unity and the CNs of class III kernels are numerically very high (approaching infinity) due to the presence of zeros in the Fourier domain. However, class II kernels have high finite CNs, around 10^4. We can observe these in Fig. 40. The range of CNs of the optimized kernels is within one order of magnitude despite being pre-pended to different neural networks and trained on different data sets.

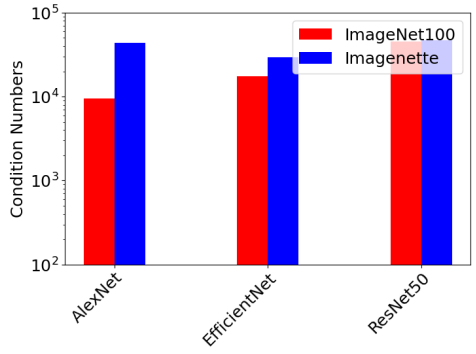


Figure 40. Condition numbers of post-optimization class II kernels pre-pended to different neural networks and trained on ImageNet-100 and ImageNette. Each bar is a mean of up to 5 runs.

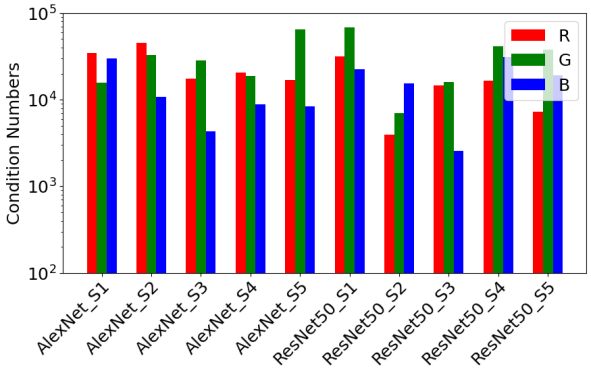


Figure 41. Condition numbers of post-optimization class II kernels pre-pended to networks specified in the xlabel and trained on ImageNette. S1-S5 refers to different seed values, and R, G, and B refer to the color channels.

11. Extended discussion

In the main paper, we only briefly visit two interesting discussion points, which we pick up here to extend the discussion.

11.1. Static Low Pass Filtering

The first point arises when considering Table 4, in particular the results for the non-trainable, static Gaussian kernel. This kernel defines a particular subspace projection, which one would expect to perform well under high-frequency noise. Yet, it also removes information in a fixed and predetermined way which leads to a decay in the clean classification accuracy, which becomes more severe as the hardness of the considered classification task increases. While in ImageNette, the static Gaussian kernel only performs one percent point below a fully learned filter of the same size of 25×25, the gap is severe when looking at ImageNet-100 in Figure 42. As the classes become more difficult to discriminate, the simple Gaussian blurring yield very unsatisfactory results whereas the proposed fully trainable filter increases model robustness across various corruptions.

11.2. Sparse Coding

The second point is an extended discussion on the relationship of our approach to sparse coding. In the main paper, we started the discussion of why the standard initial layers of the usual networks, do not learn similar patterns as our proposed dimension-preserving large kernel convolution. These layers are known to learn an overcomplete representation of the data in a sparse coding sense, *i.e.* in every channel of every feature map, only very few features are "active". When considering for example a convolution layer with 3×3 kernels, it is also clear that the usual increase in feature map channels from 3 in the input to *e.g.* 64 likely leads to redundant information (a 3×3 filter can

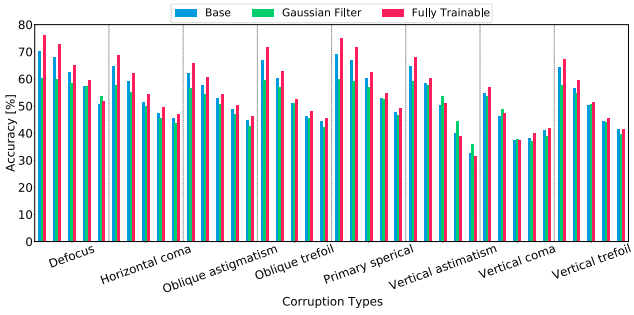


Figure 42. ResNet ImageNet100 OpticsBench. This evaluation demonstrates the benefit of our proposed fully trainable large kernel filter over a fixed Gaussian filter. While the performance gap on ImageNette is smaller, on ImageNet-100, the Gaussian prefiltered images can not be reliably classified.

be represented by 9 basis vectors, *i.e*. more than 9 feature map channels imply redundancy, not only with respect to the input information but also in the amount of different (*i.e*. linearly independent) features that can be extracted). Consequently, such usual early layers are likely to represent sparse codes, representing every kind of data that is given in the input. Such behavior is likely to impact a model's robustness, as examplified in the below toy example:

Suppose the input data lives in an approximate subspace: the clean data forms (noisy) samples of this subspace since it is only approximate. If this approximate subspace is mapped to higher dimensions via a convolutional input layer with many redundant output channels, more degrees of freedom are available to fit the data points. The optimizer may therefore choose to use additional redundant dimensions to marginally increase the fit of the noisy data points. While this improves performance on the clean data, the extrapolation capabilities to unseen samples may decrease, giving rise to overfitting and by implication worse performance on corrupted data. Forcing the optimizer to stay in the original subspace and choosing a more constrained model therefore may aid extrapolation, *i.e*. generalization capabilities of the resulting model.

In our layer, such behavior is avoided, since we are mapping every channel of the input to exactly one channel in the output of our layer. If a particular piece of information from the input is to be preserved in the output of our layer, the large kernel with its $3 \times 25 \times 25$ weights has to be learned appropriately. As discussed in section 3 **Content Preserving Filters**, only a limited set of special filters can fully preserve the input content. Yet, for the kernel to be learned, it uses gradient signal that is provided from the classification task at hand. In particular, if an input feature is not contributing to the discrimination between given classes, there will be no gradient signal provided to the input that encourages to learn this particular feature, and since the representation learned is not over-complete, it will also not be learned by chance. A large kernel makes it particularly hard to learn local patterns by chance. It learns to predominantly represent signal that is *needed* for the task at hand, *i.e*. the *Essential*.

One potential interpretation is to understand the trained layer as an information bottleneck, through which only such information is passed, that is explicitly needed for the classifier to perform well on the training set. However, removing the additional parts of the signal that are not helpful for classification makes the model more robust under input corruptions.