Distribution Mismatch Correction for Improved Robustness in Deep Neural Networks

Alexander Fuchs, Christian Knoll, Franz Pernkopf
Signal Processing and Speech Communication Laboratory
Graz University of Technology
fuchs@tugraz.at

Abstract

Deep neural networks rely heavily on normalization methods to improve their performance and learning behavior. Although normalization methods spurred the development of increasingly deep and efficient architectures, they also increased the vulnerability with respect to noise and input corruptions. In most applications, however, noise is ubiquitous and diverse; this can often lead to complete failure of machine learning systems as they fail to cope with mismatches between the input distribution during training- and test-time. The most common normalization method, batch normalization, reduces the distribution shift during training but is agnostic to changes of the input distribution during test time. Sample-based normalization methods can correct linear transformations of the activation distribution but cannot mitigate changes in the distribution shape; this makes the network vulnerable to distribution changes that cannot be reflected in the normalization parameters. We propose an unsupervised non-parametric distribution correction method that adapts the activation distribution of each layer. This reduces the mismatch between the training and test-time distribution by minimizing the 1-D Wasserstein distance. In our experiments, we empirically show that the proposed method effectively reduces the impact of intense image corruptions and thus improves the classification performance without the need for retraining or fine-tuning the model. An extended version of this paper can be found at https://arxiv.org/abs/2110.01955.

1 Introduction

Early on, Neural Networks (NNs) have proven to excel at interpolating between training data points but to fail when extrapolating to regions not covered by the training data [2, 8]. The lack of sufficiently large datasets, therefore, limited the application of NNs to tasks with well-known input distributions; this prevents any unpredictable behavior in the network that might stem from data with unknown distribution. More precisely – given a model with parameters θ, output $\mathbf{y}$, input $\mathbf{x}$, and a static conditional probability $p(\mathbf{y}|\mathbf{x}, \theta)$ – the problem of evaluating samples $\mathbf{x}$ from a different distribution than the training distribution is known as *covariate shift* [21, 24]. In recent years, the rise of big data and data augmentation techniques have alleviated the problem of distribution shifts via increasing the number of samples from the input space [13]. The problem of covariate shift, however, remained – albeit in a different form: when training Deep Neural Networks (DNNs), each parameter update causes a distribution shift for the next mini-batch in the consecutive layers, resulting in convergence problems for DNNs. In particular with DNNs the problem is further exacerbated by a large number of layers since the distribution shifts can occur internally (within the model) before every layer. Thus, the main impact of the covariate shift moved from test-time to training-time.

The introduction of *Batch normalization* (BN) reduced such internal covariate shifts during training by matching the distribution of activations across batches and, in doing so, greatly improved the

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

convergence of deep Convolutional Neural Networks (CNNs) [10]. BN, which is the most commonly used normalization method, estimates the normalization parameters, i.e. the mean and variance, based on the training set distributions. This makes BN inherently vulnerable to changes that are e.g. caused by image corruptions [3] and forces the models to extrapolate to regions not covered during training. To improve the corruption robustness of DNNs, data augmentation is used to improve robustness against specific types of corruption, but this concurrently reduces robustness against other types of corruptions [3, 7]. This highlights the importance of mitigating covariate shifts during test-time. *Group Normalization* (GN) and *Filter Response Normalization* (FRN) have been proposed to overcome the batch size dependence of BN (caused by insufficient statistics for small batches). Both methods are more flexible than BN as they compute the normalization parameter over individual samples and are thus more robust against changes in the activation distributions. One important limitation of all the above normalization methods is that they can only correct for linear distribution transformations (e.g. mean shift or variance scaling) but not for mismatches between the shape of the distribution.

We propose a non-parametric distribution correction method that utilizes the 1D-Wasserstein distance and reduces distribution mismatches of arbitrary form during test-time. This correction method can be combined with other normalization methods and corrects for changes in the distribution shape in an unsupervised setting without the need for retraining or fine-tuning of the models, in contrast to self-supervised methods which adapt at least some of the model parameters [14, 20, 25, 26]. Our proposed approach is an iterative procedure following an energy minimization scheme as used in image denoising [5, 17, 19, 28]. It is agnostic to the specific type of noise and maps all noisy activations to the target distribution of each layer. We compare the target to the test-time distribution after each activation layer and – if necessary – calculate corrections throughout the model. Given that the target distribution is based on the training data, this effectively moves the test samples closer to data points seen during training and in doing so reduces the covariate shift. Consequently, the network can better process the features in subsequent layers. A subsequent step minimizes the difference to the original activation maps again and ensures that the proposed correction does not induce unwanted distortions. In our experiments, we empirically show that our proposed method improves robustness against high-intensity noise of input corruptions. We evaluate and compare three normalization methods, i.e., BN, GN, and FRN on corrupted ImageNet (ILSVRC 2012). Furthermore, we exemplary analyze the convergence behavior of the proposed correction.

2 Related work

Many methods – such as weight regularization or dropout – aim to improve robustness in deep learning models [4, 23]. Alternatively, robustness can be improved by increasing the coverage of the input space with data augmentation. Therefore training samples are augmented by the application of affine transformations or expected noise types and are then explicitly included in the training set [1, 6, 13, 22]. But, as mentioned before, improving robustness to one corruption type via data augmentation can lead to a decrease of robustness against others [3, 7]. Other ways of improving robustness and prediction stability are representation learning techniques or capsule networks. These approaches try to learn equivariant representations of features; i.e., conceptual representations independent of the position, orientation or context [11, 12, 18, 27]. Moreover, the choice of activation functions impacts the robustness of the network as well [15, 16]. Recently, there has also been increasing interest in improving the robustness of normalization methods. The influence of input corruptions on networks using batch normalization has been investigated in [3]; this further led to a domain adaption method for the normalization that improves the robustness of DNNs against corruptions. A similar approach was taken by [20], which adapts the normalization parameters based on test-time statistics. Other approaches use self-supervised methods to retrain the model based on test samples to reduce domain shifts [14, 25, 26].

3 Distribution correction for Deep Neural Networks (DNNs)

Machine learning and signal processing tasks frequently experience performance degradation caused by noise. As noise comes in miscellaneous forms, it is challenging to achieve general robustness against arbitrary noise. Most existing normalization approaches use parametric approximations (e.g. Gaussians), and can therefore only reflect changes in the distribution parameters (i.e., the mean and

variance). Therefore, they often struggle in practice as they cannot correct for shape mismatches in the activation distribution. In order to mitigate the distribution mismatches of the test-time activations $\mathbf{a}$, we must find an effective way to suppress noisy activations while maintaining the classification performance in the subsequent layers. Therefore, we will formulate this problem as a probabilistic denoising problem. For our considerations, we recast the maximum a-posteriori problem into an equivalent energy minimization problem to simplify the optimization procedure. The energy $E(\tilde{\mathbf{a}}|\mathbf{a})$ is composed of two terms $\mathcal{R}(\tilde{\mathbf{a}})$ (corresponding to the prior) and $D(\mathbf{a}|\tilde{\mathbf{a}})$ (corresponding to the likelihood) so that

$$E(\tilde{\mathbf{a}}|\mathbf{a}) = \lambda_2 \mathcal{D}(\mathbf{a}|\tilde{\mathbf{a}}) + \lambda_1 \mathcal{R}(\tilde{\mathbf{a}}), \tag{1}$$

where $\tilde{\mathbf{a}}$ represents the corrected activations. This type of problem can be solved iteratively alternating between prior and likelihood term. Note that we choose independent values for the parameters of the prior-update, i.e., λ_1, and of the likelihood-update, i.e., λ_2, that implicitly determine the relative importance of the corresponding terms. For this form, we must, at the one hand, specify a suitable prior term $\mathcal{R}(\tilde{\mathbf{a}})$ that reduces the covariate shift in each layer without restricting the network. The data likelihood term $\mathcal{D}(\mathbf{a}|\tilde{\mathbf{a}})$, on the other hand, preserves the spatial correlations of the activation maps and prevents the loss of valuable information. By minimizing both terms jointly, one can achieve an optimal trade-off between minimizing the covariate shift and representing the available data. Further implementation details are presented in the pseudocode in Appendix B.

Typically, parametric distributions do not provide good representations of the activation distributions in DNNs. Consequently, any correction method that approximates the prior by a parametric target distribution $q_\theta(\mathbf{a})$ distorts the shape of the true activations distribution $p(\mathbf{a})$. Therefore, to reduce distortions, $q(\mathbf{a})$ must be non-parametric as well. The Wasserstein distance proves to be particularly well-suited for our correction method: not only does it allow us to effectively minimize the mismatch between the distributions during training- and test-time, but it also provides an elegant way of representing a non-parametric distribution in one dimension.

To find a well-suited target distribution, we must first sort the N activations $\mathbf{a}$ for each sample m in the training set in ascending order. We flatten the channels and create a single distribution across the height H, width W, and channel C dimension of the layer resulting in $N = H \times W \times C$ activation values. Second, to minimize the non-parametric prior term, we need to calculate the Wasserstein distance between the activations distribution during test-time (i.e., $p(\mathbf{a})$) and the target distribution $q(\mathbf{t})$ obtained from the training set.

The one-dimensional Wasserstein distance between $q(\mathbf{t})$ and $p(\mathbf{a}^{(m)})$ is given according to

$$W\big(p(\mathbf{a}^{(m)}), q(\mathbf{t})\big) = \left(\sum_{i=1}^{N} || \Big(a_{(i)}^{(m)} - \frac{1}{N} \sum_{i=1}^{N} a_{(i)}^{(m)} \Big) - t_{(i)} ||^r \right)^{\frac{1}{r}}, \tag{2}$$

where $t_{(i)}$ are the sorted target values calculated using the Wasserstein barycenter of the training set and $a'^{(m)}_{(i)}$ are the sorted test-time activations. Here we subtract the mean of the activations, as we do not care about the precise location of the distribution but primarily about its shape. Let $r = 1$; then, the Wasserstein distance between $p(\mathbf{a})$ and $q(\mathbf{t})$ from (2) is minimized by updating the (unsorted) activation with index j according to $\Delta_j = t_{(i)} - a_{(i)}$. We apply the correction after the ReLU activation function; thus many activations are zero. Our updates must preserve this sparsity as the performance will degrade otherwise. Therefore, we explicitly enforce sparsity in the prior term $\mathcal{R}$, i.e., we prevent correcting activations with $a = 0$ by adding an infinitely deep energy-well $-\delta(a)$, where $\delta(\cdot)$ denotes the Dirac delta.[1] Combining this sparsity term with the Wasserstein distance finally leads to the following prior term:

$$\mathcal{R}(\tilde{\mathbf{a}}) = W\big(p(\mathbf{a}'^{(m)}), q(\mathbf{t})\big) - [\delta(a_i)]. \tag{3}$$

Minimizing only the prior term might have undesired side effects and destroy important structure in the channels of the network, i.e the spatial correlation. Therefore, the energy minimization needs to find a trade-off between matching the distributions and conserving the spatial correlation. We achieve this by considering a likelihood term $\mathcal{D}(\mathbf{a}|\tilde{\mathbf{a}})$, modeled by

$$\mathcal{D}(\mathbf{a}|\tilde{\mathbf{a}}) = \frac{1}{2} ||\mathbf{a} - \tilde{\mathbf{a}}||^2 \tag{4}$$

that conserves the structure in the data. This expression is also straighforward to minimize.

[1] Note that this sparsity constraint is not required if the correction is applied directly after the convolution.

4 Experiments

In our experiments we first analyze the convergence behavior of the correction algorithm on corrupted MNIST and present results for the corrupted classification ILSVRC 2012 dataset (ImageNet-C) using three different normalization methods (BN, FRN, GN) for the models.

Convergence behavior As our correction method is applied iteratively, we investigate its convergence behavior for the average classification accuracy on the corrupted MNIST dataset. Here we vary the step size parameters λ_1 and λ_2 and show their influence. In Figure 1 we see that BN, FRN, and

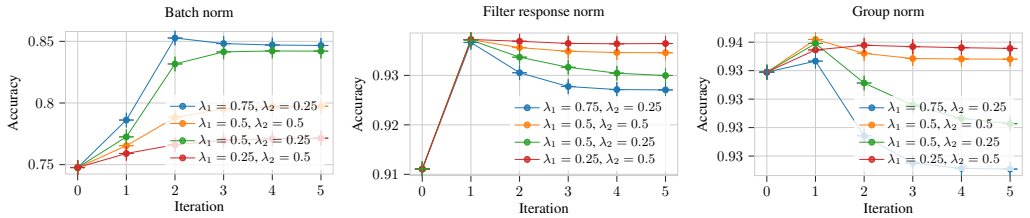

Figure 1: Performance over the number of used iterations for the correction algorithm for a single model trained on MNIST. Each normalization method uses 4 sets of step size parameters λ_1 and λ_2.

GN, show significantly different behavior for the same choices of λ_1 and λ_2. FRN and GN obtain substantially better results than BN even without using the correction (at iteration 0). Also, BN is the only model that can substantially improve its classification performance by running the algorithm for more than 1 iteration. FRN shows consistent results after one iteration, whereas the performances diverge for more iterations. GN also shows improvement within the second iteration for the parameter set $\lambda_1 = 0.25,\ \lambda_2 = 0.5$, but does not outperform the best model obtained with only one iteration.

Corrupted ImageNet classification The corrupted ILSVRC 2012 dataset contains 19 different corruption variants of the original dataset. It contains 50000 RGB images of size 224×224 per corruption that we used for our evaluations. For the evaluation on the corrupted ImageNet (ILSVRC 2012) dataset we again choose the same parameter set $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$ and $N_{iter} = 1$ for all normalization methods. We trained a single ResNet-50 model for 90 epochs using an SGD optimizer with a base learning rate of 0.1 on the clean ImageNet dataset. The learning rate was decayed after 30, 60, and 80 epochs, and a warm-up from 0.02 to 0.1 was used during the first five epochs of training.

Table 1: Classification performance in % on the ImageNet-C dataset over all 5 severities. (c) indicates the model with our distribution correction enabled. The mean Corruption Error (mCE) is calculated using a AlexNet baseline [9].

noise type	BN	BN (c)	FRN	FRN (c)	GN	GN (c)
avg. top 1 accuracy	44.24 %	45.15 %	43.72 %	44.80 %	47.27 %	**48.20** %
mCE	70.95 %	69.93 %	71.75 %	70.49 %	67.35 %	**66.23** %

The results in Table 4 show that for the corrupted ImageNet dataset, all models regardless of the used norm achieve about 1% performance improvement by using the proposed correction method. Generally, we see that the more flexible GN models are the most robust against image corruptions outperforming BN and FRN by >3.0%. This supports our assumption that even for large amounts of data, covariate shift still influences performance for corrupted inputs.

5 Conclusion and Outlook

We proposed a non-parametric activation distribution correction method based on the Wasserstein distance. It reduces the mismatch between test-time and training distributions of the activations within DNNs. The proposed method uses a maximum a-posteriori estimate, determined by minimizing the energy with respect to a data likelihood term and a prior term based on the Wasserstein distance. Our proposed method works in an unsupervised setting and can be retrofitted into existing networks without retraining. In our experiments, we showed that our correction algorithm can effectively reduce the mismatch between test-time and training distributions. This improves classification performance on corrupted input data. For future applications, we want to extend our robustness evaluations and explore the use of our algorithm for reducing the impact of parametric approximations.

References

[1] H. S. Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

[2] E. Barnard and L. Wessels. Extrapolation and interpolation in neural network classifiers. *IEEE Control Systems Magazine*, 12(5):50–53, 1992. doi: 10.1109/37.158898.

[3] P. Benz, C. Zhang, A. Karjauv, and I. S. Kweon. Revisiting batch normalization for improving corruption robustness. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 494–503, January 2021.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[6] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[7] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk. Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning*, pages 2280–2289. PMLR, 2019.

[8] P. Haley and D. Soloway. Extrapolation limitations of multilayer feedforward neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 25–30 vol.4, 1992. doi: 10.1109/IJCNN.1992.227294.

[9] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[11] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673, 2020.

[12] D. H. Kim, W. J. Baddar, J. Jang, and Y. M. Ro. Multi-objective based spatio-temporal feature representation learning robust to expression intensity variations for facial expression recognition. *IEEE Transactions on Affective Computing*, 10(2):223–236, 2017.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[14] J. Liang, D. Hu, and J. Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.

[15] D. Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4, 2019.

[16] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[17] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[18] F. D. S. Ribeiro, G. Leontidis, and S. Kollias. Capsule routing via variational Bayes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3749–3756, 2020.

[19] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[20] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

[21] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[22] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition*, volume 3, pages 958–958. IEEE Computer Society, 2003.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 (56):1929–1958, 2014.

[24] M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[25] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[26] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=uXl3bZLkr3c`.

[27] C. Zhou and R. C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674, 2017.

[28] M. Zhu and T. Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. Technical report, 2008.

A Analyzing the distributions

Since we assume a quasi-stationary distribution of the activations of the flattened layer across the training set, we need to verify if this assumption holds for our ResNet models. This is necessary in order to create a meaningful target distribution $q(\mathbf{t})$. Therefore, we analyze the variance σ^2 of the target values $\mathbf{t}$ over the training set:

$$\sigma^2_{(i)} = \frac{1}{M} \sum_{m=1}^{M} (t_{(i)} - a^{(m)}_{(i)})^2. \tag{5}$$

In Figure 2, we see that although all normalization methods have a similar shape for the variance of

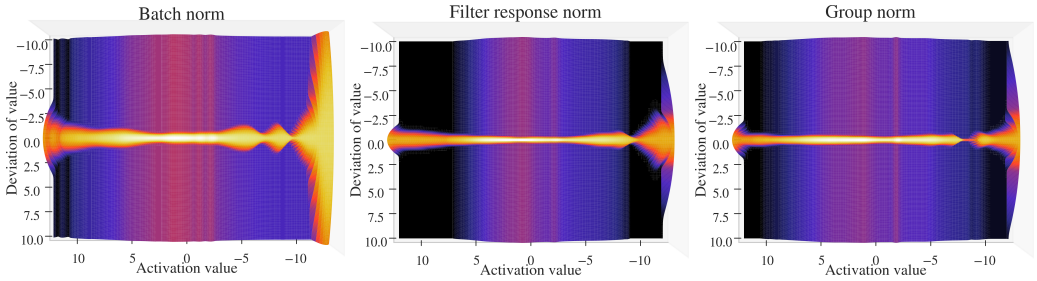

Figure 2: The log probability distributions $\log(p(x_{(i)}|t_{(i)}, \sigma^2(t_{(i)})))$ of a sorted value $(x_{(i)})$ given the corresponding target values $t_{(i)}$ and variances $\sigma^2(t_{(i)})$, assuming Gaussian distributed target values. Here $\log(p(x_{(i)}|t_{(i)}, \sigma^2(t_{(i)})))$ is shown for the different normalization methods BN, FRN and GN for activations before the first ReLU function. The x-axis shows the mean value of the i^{th} sorted target value and the y-axis shows the deviation from the mean over the training set.

their target value distribution, BN experiences a much higher variation at the tails of the distributions, having a value range 5 times higher than GN and FRN. This observation is in line with the intuition that more flexible normalization methods produce narrower activation distributions.

Generally, we see that with the exception of the distribution-tails, the distribution of the targets is more or less stationary. This is an interesting observation, as one would expect that the activation distributions for the entire layer have a higher dependency on the specific input. Following this observation, we analyze this behavior by comparing the activation distributions for the most dissimilar examples in the test set. Therefore we calculate the Wasserstein distances from a test example k to all other test examples m with respect to their channel distributions, utilizing the order statistics for each channel j separately and select the sample with the largest total value with index $\tilde{m}$,

$$\tilde{m} = \arg\max_{m} \sum_{i,j} |a^{(k)}_{j,(i)} - a^{(m)}_{j,(i)}|. \tag{6}$$

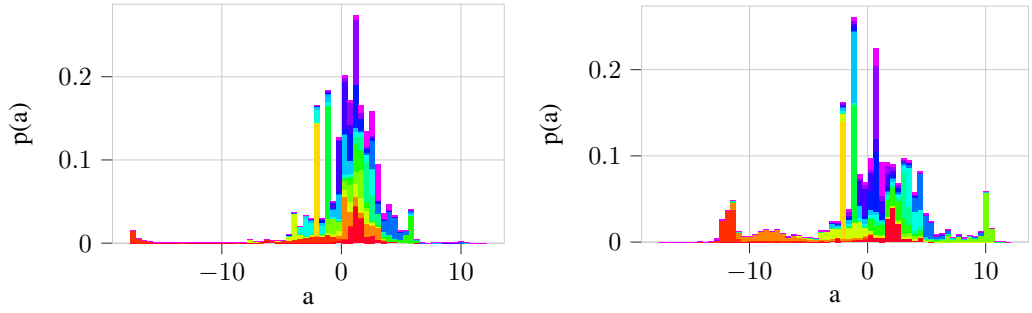

Figure 3: The distribution over all channels of the first layer of a ResNet-20 classifier trained on CIFAR-10 before the ReLU activation. Both images shows the distribution of activations over all channels stacked, indicated by using a different color for each channel. The left image is the test sample k for which the image with the largest Wasserstein distance within the entire test set (with respect to channel distributions) was selected. The most dissimilar image $\tilde{m}$ is shown on the right.

Figure 3 shows that while the individual channel distributions are very distinct, the shape of the distribution over all channels is more or less stable. This indicates that the inverse correlation of channels can conserve the overall shape of the distribution for each layer.

B Algorithm

The algorithm is described in the following pseudo-code. The algorithm is executed for every layer and returns the corrected activations $\tilde{\mathbf{a}}$. The step-sizes λ_1 and λ_2 determine the relative importance of the prior term versus the likelihood term.

Algorithm 1 Activation Correction Algorithm

Input: Unsorted activations $\mathbf{a}$, sorted target values $\mathbf{t}$, step-sizes λ_1, λ_2, and maximum number of iterations N_{iter}

Output: Corrected activations $\tilde{\mathbf{a}}$

$N \leftarrow \text{len}(\mathbf{a})$
$\tilde{\mathbf{a}} \leftarrow \mathbf{a} - \frac{1}{N}\sum_i a_i$
for $n < N_{iter}$ **do**
  $[a_{(i)}], \mathbf{j} \leftarrow \text{sort}(\tilde{\mathbf{a}})$
  for $i < N$ **do**
    $j \leftarrow j_i$
    $\tilde{a}_j \leftarrow \tilde{a}_j + \lambda_1 \cdot (t_{(i)} - a_{(i)})$
    $\tilde{a}_j \leftarrow \tilde{a}_j + \lambda_2 \cdot (a_j - \tilde{a}_j)$
  end for
end for
for $j < N$ **do**
  if $a_j \neq 0$ **then**
    $\tilde{a}_j \leftarrow \tilde{a}_j$
  else
    $\tilde{a}_j \leftarrow a_j$
  end if
end for

C Supplemental material of experiments

Table 2: Classification accuracy in % on the corrupted ImageNet dataset with corruption severity 5. (c) indicates the model with the distribution correction enabled.

noise type	BN	BN (c)	FRN	FRN (c)	GN	GN (c)
brightness	58.15 %	56.60 %	57.71 %	58.11 %	**58.32** %	58.14 %
contrast	5.38 %	8.63 %	5.71 %	12.24 %	26.39 %	**30.12** %
defocus blur	15.70 %	14.49 %	14.57 %	15.95 %	14.43 %	**15.63** %
elastic	14.90 %	16.85 %	18.31 %	18.58 %	20.00 %	**20.55** %
fog	44.66 %	47.85 %	39.35 %	43.25 %	49.99 %	**50.94** %
frost	26.48 %	29.99 %	26.45 %	30.09 %	28.09 %	**31.28**%
gaussian blur	**12.97** %	10.99 %	11.68 %	12.83 %	11.50 %	12.30 %
gaussian noise	4.04 %	7.19 %	4.48 %	5.71 %	9.74 %	**10.98** %
glass blur	7.78 %	9.01 %	9.94 %	**10.07** %	8.54 %	9.02 %
impulse noise	4.36 %	8.07 %	5.19 %	6.79 %	10.70 %	**12.05** %
jpeg compression	**42.73** %	41.5 %	30.64 %	31.22 %	39.58 %	39.14 %
motion blur	9.29 %	11.20 %	13.83 %	14.95 %	14.27 %	**14.99** %
pixelate	51.04 %	50.72 %	49.28 %	50.05 %	51.57 %	**54.26** %
saturate	49.22 %	50.62 %	51.48 %	51.35 %	**53.18** %	52.97 %
shot noise	5.57 %	8.59 %	6.27 %	7.84 %	10.86 %	**12.08** %
snow	20.65 %	22.26 %	26.98 %	28.89 %	27.19 %	**29.38** %
spatter	30.49 %	32.38 %	32.64 %	34.56 %	35.27 %	**36.38** %
speckle noise	16.01 %	19.56 %	19.98 %	18.06 %	22.63 %	**23.80** %
zoom blur	24.14 %	**25.10** %	22.68 %	23.78 %	23.94 %	24.85 %
average	23.53 %	24.82 %	23.43 %	25.06 %	27.17 %	**28.38** %