Mix-MaxEnt: Improving Accuracy and Uncertainty Estimates of Deterministic Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose an extremely simple approach to regularize a single deterministic neural network to obtain improved accuracy and reliable uncertainty estimates. Our approach, on top of the cross-entropy loss, simply puts an entropy maximization regularizer corresponding to the predictive distribution in the regions of the embedding space between the class clusters. This is achieved by synthetically generating between-cluster samples via the convex combination of two images from *different* classes and maximizing the entropy on these samples. Such a data-dependent regularization guides the maximum likelihood estimation to prefer a solution that (1) maps out-of-distribution samples to high entropy regions (creating an entropy barrier); and (2) is more robust to the superficial input perturbations. We empirically demonstrate that Mix-MaxEnt consistently provides much improved classification accuracy, better calibrated probabilities for in-distribution data, and reliable uncertainty estimates when exposed to situations involving domain-shift and out-of-distribution samples.

1 Introduction

A particularly thriving sub-field of research in Deep Neural Networks (DNNs) concerns devising efficient approaches towards obtaining reliable predictive uncertainty. NNs are known to be overconfident for both, in- and out-distribution samples (i.e. for samples coming from the same distribution from which the training distribution has been sampled (IND samples) and for samples not coming from such distribution), leading to highly unreliable uncertainty estimates [Guo et al., 2017, Taori et al., 2016, Ovadia et al., 2019]. The overconfidence problem becomes even more concerning when just slight changes in illumination, atmospheric conditions or in the image capturing process (domain-shift) can severely damage the actual accuracy of the model [Taori et al., 2016]. A desirable property of any model is to be robust to such superficial changes that do not affect the label of the classified image, and to become uncertain (or indecisive) when exposed to samples from a distribution different from the training distribution.

Based on our analyses and observations about how out-of-distribution (OOD) and data-shifted (DS) inputs are mapped by Neural Networks, we design a simple entropy maximization regularizer (called *Mix-MaxEnt*) enforced for samples *synthesized* using the convex combination of a pair of samples from two **different** classes of the in-distribution training data. When combined with the cross-entropy loss, this regularizer prefers a maximum likelihood solution such that the uncertainty of the network increases while moving away from the embeddings of one class in the direction of another, and learns features that are robust to local input perturbations. Through extensive experiments using WideResNet28-10 and ResNet50 architerctures on CIFAR10 and CIFAR100 datasets, we demonstrate that our method outperforms *all* existing single model baselines in providing clean data accuracy. On Domain-Shift experiments, it provides remarkably improved accuracy compared to all the baselines including Deep Ensembles (DE) [Lakshminarayanan et al., 2017].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

2 Analysis and observations

Consider the supervised task of learning a predictive distribution $p(y|\mathbf{x};\theta)$, where $y \in \{1, \cdots, K\}$ is the set of K-classes, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$ is the space of input data, and θ respresents the parameters of the underlying model (*e.g.* a neural network). To obtain an optimal θ, the standard approach is to collect a training data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, and maximize the log-likelihood $p(y|\mathbf{x} \in \mathcal{X};\theta)$ under $\mathcal{D}$. Therefore, the nature of the predictive distribution heavily relies on the training data and the model under consideration. In practice, however, because of the high dimensionality of the input space $\mathcal{X}$, the training data $\mathcal{D}$ is normally collected from a subset $\mathcal{X}_I \subset \mathcal{X}$, where $\mathcal{X}_I$ denotes the in-domain data manifold. Let us denote the out-domain data manifold as $\mathcal{X}_O = \mathcal{X} \setminus \mathcal{X}_I$.

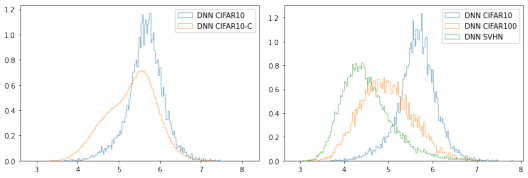

Figure 1: Histograms of the Euclidean distance of the embeddings (model trained on CIFAR-10). (Left) CIFAR-10 vs CIFAR-10-C. (Right) CIFAR-10 (IND) vs CIFAR-100 (OOD) and SVHN (OOD). Note, both corrupted and OOD embeddings lie within the hypersphere $\mathcal{S}$ as their histograms are left shifted relative to the histogram of CIFAR-10. Also, the SVHN histogram is more shifted towards the center than that of CIFAR-100, showing an implicit ordering.

Where do neural networks map $\mathcal{X}_O$? Modern softmax-based neural classifiers operate under two essential assumptions: (1) a closed-world assumption [Bendale and Boult, 2015]; (2) an implicit clutering assumption [Hess et al., 2020, Lee et al., 2018]. The closed-world assumption implies that the classifier must select one out of K classes fixed at training time, even if the input belongs to none of them. The clustering assumption depends on the presence of the softmax layer: the authors of [Hess et al., 2020] have shown that softmax neural classifiers perform K-means clustering (which, we recall, can be interpreted as a limit subcase of Expectation-Maximisation under Gaussian Mixture Model (GMM) assumptions [Hastie et al., 2001, Bishop, 2006]) with K centroids at equal distance from the center of the embedding space. The authors of [Lee et al., 2018] observed the embeddings of modern neural network do empirically adapt well to GMM assumptions.

Following the above insights, we consider a WideResNet (WRN) trained on CIFAR-10 ($\mathcal{X}_I$) and obtain the smallest hypersphere $\mathcal{S}$ in the feature space that contains all the in-domain test samples. Note, as already mentioned in Hess et al. [2020], we also found the mean of the embeddings of $\mathcal{X}_I$ to be very close to a zero vector, therefore, $\mathcal{S}$ is practically centred at zero. Given $\mathcal{S}$, we would like to understand where do $\mathcal{X}_O$ (CIFAR-100 and SVHN) and data-shifted samples (corrupted CIFAR-10 test samples denoted as CIFAR-10-C) lie with respect to $\mathcal{S}$. We empirically notice the NN maps all (except one) out-of-distribution ($\mathcal{X}_O$) and data-shifted samples (corrupted) inside the hypersphere $\mathcal{S}$ that was obtained using the test data belonging to $\mathcal{X}_I$, as shown in Figure 1. This implies OOD and DS samples are either mapped closer to the center of the embedding space, in the space between the class clusters or are projected within the class clusters.

Forcing classifiers to be uncertain where it truly matters Recall that an ideal classifier's predictive distribution $p(y|\mathbf{x} \in \mathcal{X};\theta)$ would assume the following form

$$\underbrace{p(y|\mathbf{x} \in \mathcal{X}_I;\theta)p(\mathbf{x} \in \mathcal{X}_I)}_{\textit{maximum likelihood estimate}} + \underbrace{p(y|\mathbf{x} \in \mathcal{X}_O;\theta)p(\mathbf{x} \in \mathcal{X}_O)}_{\textit{unknown during training}}.$$

Since $\mathcal{X}_O$ is unknown during the maximum likelihood estimation (MLE) of θ, the model is completely unaware of the second part of the predictive distribution that depends on the presence of $\mathcal{X}_O$. Therefore, in order for the model to be indecisive for out-of-distribution and domain-shifted samples (assuming they are away from the in-distribution data), it is desirable that the model's uncertainty increases proportionally to the test sample distance from the training data $\mathcal{X}_I$.

Given the observations from our analysis, we find that the OOD and domain-shifted samples are mapped to a specific embedding space $\mathcal{S}$ where the network tends to concentrate all its embeddings. This indicates that correcting its uncertainty estimates within this space $\mathcal{S}$ could be sufficient to achieve better performance, rather than trying to increase entropy everywhere away from the training data (i.e. mostly in regions where the network never projects its inputs). This is precisely the motivation behind our work. In what follows, we present an extremely simple approach to regularize the embedding space $\mathcal{S}$ such that entropy barriers are being created between different classes.

	Clean Data			Domain-Shift			Out-of-Distribution			
	CIFAR100 (Test)			CIFAR100-C			CIFAR10		SVHN	
Methods	**Accuracy** (↑)	**ECE** (↓)	**AdaECE** (↓)	**Accuracy** (↑)	**ECE** (↓)	**AdaECE** (↓)	AUROC (↑)	AUPR (↑)	AUROC (↑)	AUPR (↑)
DNN	81.58 ± 0.13	3.88 ± 0.25	3.84 ± 0.24	52.54 ± 0.31	9.96 ± 0.21	9.94 ± 0.21	81.06 ± 0.29	77.35 ± 0.39	79.68 ± 4.81	88.46 ± 2.53
DNN-SN	81.60 ± 0.15	3.94 ± 0.23	3.81 ± 0.21	52.61 ± 0.23	11.62 ± 0.41	11.59 ± 0.41	81.10 ± 0.35	77.34 ± 0.19	83.43 ± 3.63	91.01 ± 2.05
DNN-SRN	81.38 ± 0.23	3.82 ± 0.27	3.71 ± 0.26	52.54 ± 0.17	11.04 ± 0.77	11.00 ± 0.78	81.26 ± 0.18	77.36 ± 0.30	85.51 ± 1.18	91.84 ± 1.12
Deep Ensembles	$\mathbf{83.85 \pm 0.13}$	3.31 ± 0.12	3.29 ± 0.08	55.58 ± 0.14	12.43 ± 0.13	12.36 ± 0.15	$\mathbf{83.26 \pm 0.14}$	$\mathbf{79.82 \pm 0.27}$	85.07 ± 1.58	91.65 ± 0.97
SNGP	79.20 ± 0.21	1.95 ± 0.25	1.94 ± 0.28	57.23 ± 0.25	10.45 ± 1.56	10.43 ± 1.56	79.05 ± 0.29	75.09 ± 0.34	86.78 ± 1.90	93.30 ± 1.05
KFAC-LLLA	81.56 ± 0.07	2.20 ± 0.31	2.30 ± 0.32	52.57 ± 0.27	8.97 ± 0.21	8.99 ± 0.21	81.04 ± 0.35	77.36 ± 0.34	80.32 ± 4.41	89.05 ± 2.30
Mixup	82.60 ± 0.37	1.77 ± 0.49	1.98 ± 0.43	56.99 ± 0.54	10.32 ± 0.64	10.45 ± 1.57	78.37 ± 1.20	75.95 ± 0.56	78.68 ± 4.29	88.27 ± 1.89
Mix-MaxEnt	$\underline{83.23 \pm 0.22}$	$\mathbf{1.67 \pm 0.59}$	$\mathbf{1.76 \pm 0.62}$	$\mathbf{59.39 \pm 0.72}$	$\mathbf{7.93 \pm 0.84}$	$\mathbf{7.93 \pm 0.84}$	$\underline{81.04 \pm 0.48}$	77.28 ± 0.35	$\mathbf{89.32 \pm 1.61}$	$\mathbf{94.45 \pm 0.90}$

Table 1: WideResNet28-10 trained on C100. See Appendix A for the cross-validated hyperparameters.

3 Methodology

Mix-MaxEnt Motivated by the evidence we collected, we define a simple regularization approach to make the maximum likelihood estimate aware (increasing the uncertainty and becoming less confident) of what is unknown to it during the training by simply putting a maximum entropy regularizer [Pereyra et al., 2017] on synthetically generated samples that leverage the knowledge of $\mathcal{X}_I$ to drift towards $\mathcal{X}_O$. Specifically, since $\mathcal{X}_O$ is unknown during the training, we take a simple approach where we synthetically create samples $\bar{\mathbf{x}} \in \mathcal{X}$ as follows:

$$\bar{\mathbf{x}} = \lambda_0 \mathbf{x}_i + (1 - \lambda_0)\mathbf{x}_j, \;\; if \; y_i \neq y_j. \tag{1}$$

Here, $\{\mathbf{x}_i, \mathbf{x}_j\} \in \mathcal{X}_I$, and $\lambda_0 \sim \texttt{Beta}(\alpha, \beta)$, where α and β are the parameters of the beta distribution. Note, we only choose pair of samples belonging to two *different classes* $(y_i \neq y_j)$, so that $\bar{\mathbf{x}}$ will mix features mimicking an off-data manifold image that has intermediate properties between those of each class. We further ensure this by picking $\alpha = \beta >> 1$ as, in this case, the beta distribution will have the peak in the middle and the sharpness of the peak increases as α grows. Let us call the collection of such synthetic intermediate samples $\bar{\mathcal{X}}_O$. Then, our final objective function adds to the usual cross-entropy optimisation term, an entropy maximisation term on such samples:

$$\min_{\theta} -\log p(y|\mathbf{x} \in \mathcal{X}_I; \theta) - \mathcal{H}_{\bar{y}}(p(.|\mathbf{x} \in \bar{\mathcal{X}}_O; \theta)), \tag{2}$$

where, $\mathcal{H}_{\bar{y}}(.)$ is the entropy defined over the label support set $\bar{y} = \{y_i, y_j\}$. It is important to observe that since the input image will contain features coming from the inputs of the two classes, it is reasonable to maximise the entropy only over such classes while keeping the probability assigned to the other classes to zero (since no input features endorses the presence of these classes). Since we increase the entropy only for mixed (interpolated) samples from different classes, we call our approach **Mix-MaxEnt**.

4 Experimental Results

Training Datasets and Network Architectures We employ WideResNet28-10 (WRN) [Zagoruyko and Komodakis, 2016] and ResNet50 (RN50) [He et al., 2016] architectures that have been shown to produce state-of-the-art classification accuracies on real-world datasets. We train them on CIFAR-10 (C10) and CIFAR-100 (C100). For **Domain-Shift** experiments, we resort to the widely used CIFAR10-C and CIFAR100-C, corrupted versions of C10 and C100 [Hendrycks and Dietterich, 2019]. For **Out-of-Distribution** detection experiments, following SNGP [Liu et al., 2020], we use C100 and SVHN as OOD for models trained on C10. Similarly, for models trained on C100, we use C10 and SVHN as OOD. For an exaustive description of the experimental settings, refer to Appendix A.

Evaluation Metrics For calibration (i.e. measuring overconfidence or underconfidence), we employ: (1) the widely used Expected Calibration Error (ECE) [Guo et al., 2017], and (2) the recently proposed Adaptive ECE (AdaECE) [Mukhoti et al., 2020]. For all the methods, the ECE and AdaECE are computed after performing temperature scaling [Guo et al., 2017] with a cross-validated temperature parameter.

Accuracy and Calibration on Clean Data Our first set of experiments evaluates the accuracy of the proposed method on the test sets of C100 and C10. As it can be observed from the 'Clean

	Clean Data			Domain-Shift			Out-of-Distribution			
	CIFAR10 (Test)			CIFAR10-C			CIFAR100		SVHN	
Methods	**Accuracy** (↑)	**ECE** (↓)	**AdaECE** (↓)	**Accuracy** (↑)	**ECE** (↓)	**AdaECE** (↓)	AUROC (↑)	AUPR (↑)	AUROC (↑)	AUPR (↑)
DNN	96.14 ± 0.08	1.26 ± 0.05	1.34 ± 0.03	76.60 ± 0.28	12.64 ± 0.77	12.62 ± 0.77	88.61 ± 0.34	88.91 ± 0.21	96.00 ± 1.10	98.08 ± 0.66
DNN-SN	96.22 ± 0.11	0.71 ± 0.14	1.12 ± 0.16	76.56 ± 0.24	11.13 ± 0.33	11.15 ± 0.33	88.56 ± 0.36	89.01 ± 0.34	95.59 ± 0.49	97.85 ± 0.22
DNN-SRN	96.22 ± 0.10	1.24 ± 0.08	1.36 ± 0.15	77.21 ± 0.39	11.97 ± 0.40	11.96 ± 0.4	88.46 ± 0.36	88.84 ± 0.37	96.12 ± 1.61	98.10 ± 0.81
Deep Ensembles	96.75 ± 0.05	0.81 ± 0.16	1.04 ± 0.08	78.32 ± 0.06	10.11 ± 0.17	10.31 ± 0.10	$\mathbf{91.25 \pm 0.14}$	$\mathbf{91.12 \pm 0.15}$	$\mathbf{97.53 \pm 0.69}$	$\mathbf{98.84 \pm 0.28}$
SNGP	95.98 ± 0.11	0.84 ± 0.13	0.87 ± 0.16	78.37 ± 0.22	11.33 ± 0.38	11.34 ± 0.39	$\underline{90.61 \pm 0.07}$	$\underline{90.39 \pm 0.12}$	95.25 ± 0.55	97.98 ± 0.18
DUQ	94.7 ± 0.02	3.4 ± 0.2	-	71.6 ± 0.02	18.3 ± 1.1	-	-	85.4 ± 1.0	-	97.3 ± 1.0
KFAC-LLLA	96.11 ± 0.04	1.06 ± 0.08	1.12 ± 0.07	76.56 ± 0.48	11.69 ± 0.76	11.67 ± 0.76	89.33 ± 0.23	88.52 ± 0.20	94.17 ± 1.38	96.99 ± 0.94
Mixup	97.01 ± 0.11	0.94 ± 0.21	1.16 ± 0.13	81.68 ± 0.62	$\mathbf{7.54 \pm 0.83}$	$\mathbf{7.83 \pm 1.2}$	83.17 ± 0.87	85.47 ± 0.45	87.53 ± 6.07	95.08 ± 2.12
Mix-MaxEnt	$\mathbf{97.44 \pm 0.06}$	$\mathbf{0.63 \pm 0.08}$	$\mathbf{0.50 \pm 0.08}$	$\mathbf{83.10 \pm 1.48}$	$\underline{10.13 \pm 1.59}$	$\underline{10.08 \pm 1.59}$	89.13 ± 0.18	88.12 ± 0.37	96.22 ± 0.49	98.01 ± 0.41

Table 2: WideResNet28-10 trained on C10. See Appendix A for the cross-validated hyperparameters.

Data' column of Tables 1 and 2, our method provides a remarkable increase in accuracy compared to recently proposed approaches for improved uncertainty estimation. In terms of calibration, our method remarkably improves the ECE and AdaECE outperforming all the baselines.

Accuracy and Calibration on Corrupted Data (Domain-Shift) To evaluate the behaviour of various models under domain-shift, we resort to the widely used CIFAR-100-C and CIFAR-10-C datsets, corrupted versions of the C10 and C100 datasets [Hendrycks and Dietterich, 2019]. The dataset is made by applying 15 synthetically generated but realistic corruptions at 5 degrees of intensity on the test sets of C100 and C10, respectively. The desired behaviour would be to preserve the classification accuracy as much as possible as these corruptions do not impact the underlying label, and to have an appropriate reduction in the confidence when the accuracy of the model degrades so that it is not incorrect with very high confidence. We report the expected accuracy, ECE and AdaECE, averaged over all the corruptions and degrees of intensities in the column 'Domain-Shift' of Tables 1 and 2. It is evident that our approach produces a remarkable improvement in the average accuracy compared to all the baselines. Similarly, the ECE and AdaECE are greatly improved.

Performance when exposed to Out-Of-Distribution Samples Following the standard evaluation methodology [Liu et al., 2020], we report the performance in terms of Area Under Receiver Operating Characteristic (AUROC) and Area Under Precision-Recall (AUPR) curves for the binary classification problem between in- and out-distribution samples. We use the Dempster-Shafer [Sensoy et al., 2018] uncertainty metrics in the tables, except for Mixup (for which we observe Entropy to work better). Mix-MaxEnt either outperforms all other models or is a runner-up.

Mix-MaxEnt encourages compact and separated clusters We use the well known Fisher criterion [Bishop, 2006, Chapter 4] to quantify the compactness and separatedness of the feature clusters for various models. Let $\mathcal{C}_k$ denotes the indices of samples for k-th class. Then, the overall *within-class* covariance matrix is computed as $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$, where $\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\phi(\mathbf{x}_n) - \mu_k)(\phi(\mathbf{x}_n) - \mu_k)^\top$, $\mu_k = \sum_{n \in \mathcal{C}_k} \frac{\phi(\mathbf{x}_n)}{N_k}$, and $\phi(\mathbf{x}_n)$ denote the feature vector. Similarly, the *between-class* covariance matrix can be computed as $\mathbf{S}_B = \sum_{k=1}^{K} N_k (\mu_k - \mu)(\mu_k - \mu)^\top$, where $\mu = \frac{1}{N} \sum_{k=1}^{K} N_k \mu_k$, and N_k is the number of samples in k-th class. Then, the Fisher criterion is defined as $\alpha = \texttt{trace}(\mathbf{S}_W^{-1} \mathbf{S}_B)$. Note, α would be high when the within-class covariance is small and between-class covariance is high, thus, a high value of α is desirable. In Figures 5, 6 and 7 in Appendix C, we compute α over the C10 dataset with varying degrees of domain-shift. As the amount of corruption increases, α gradually decreases for all the models, indicating that the model is not able to differentiate different classes anymore and is projecting them too close to each other. This also explains why the accuracy and the calibration of all the models decreases as the domain-shift increases. However, *Mix-MaxEnt consistently provides the best* α. This also explains why Mix-MaxEnt performed so well under domain-shift.

5 Conclusion

We proposed Mix-MaxEnt, an extremely simple approach that regularizes a neural network to be uncertain in regions of the data manifold that are unknown during training. We conducted a wide range of experiments to show that Mix-MaxEnt significantly improves the reliability of uncertainty estimates of deep neural networks, while also providing a notable boost in the accuracy. A potential extension of our work regards the possibility of mixing features using more sophisticated methods.

References

Abhijit Bendale and Terrance Boult. Towards open set deep networks. November 2015.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJlrF24twB`.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Sibylle Hess, Wouter Duivesteijn, and Decebal Mocanu. Softmax-based classification is k-means clustering: Formal proof, consequences for adversarial attacks, and improvement through centroid based tailoring. January 2020.

Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. Fast predictive uncertainty for classification with bayesian deep networks, 2021. URL `https://openreview.net/forum?id=KcImcc3j-qS`.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in ReLU networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5436–5446. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/kristiadi20a.html`.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6402–6413. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf`.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, 2018.

Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *NeurIPS*, June 2020.

Zhiyun Lu, Eugene Ie, and Fei Sha. Uncertainty estimation with infinitesimal jackknife, its distribution and mean-field approximation, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, February 2018.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip HS Torr, and Puneet K Dokania. Calibrating deep neural networks using focal loss. In *NeurIPS*, 2020.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, 2019.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548, 2017. URL http://arxiv.org/abs/1701.06548.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Skdvd2xAZ.

A Sanyal, P H S Torr, and P K Dokania. Stable rank normalization for improved generalization in neural networks and GANs. In *ICLR*, 2020.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 3179–3189. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/a981f2b708044d6fb4a71a1463242520-Paper.pdf.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. July 2016.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13888–13899. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/36ad8b5f42db492827016448975cc22d-Paper.pdf.

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal Gal. Uncertainty estimation using a single deep deterministic neural network. In *ICML*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

A Experiment Details

Methods considered for comparisons We consider both deterministic and Bayesian approaches for comparison. Following [Liu et al., 2020], we also create two additional strong and simple baselines where a ResNet is enforced to be bi-Lipschitz using Spectral Normalization (SN) [Miyato et al., 2018] and Stable Rank Normalization (SRN) [Sanyal et al., 2020]. Therefore, we compare our approach with the following baselines:

- DNN: Standard deterministic neural network trained using cross-entropy loss.
- DNN-SN: DNN with SN [Miyato et al., 2018].
- DNN-SRN: DNN with SRN [Sanyal et al., 2020].
- SNGP: Spectrally Normalized Gaussian Process [Liu et al., 2020].
- DUQ: Deterministic Uncertainty Quantification [van Amersfoort et al., 2020].
- Mixup: Standard Mixup training [Zhang et al., 2018].
- KFAC-LLLA: KFAC-Laplace Last Layer Approximation [Kristiadi et al., 2020]. A method that makes a model Bayesian at test time by taking Laplace approximation of the last layer using a Kronecker-Factored approximation [Ritter et al., 2018].
- DE: Deep Ensembles [Lakshminarayanan et al., 2017] with 5 members. Note, it is almost 5x slower than all other approaches mentioned above.

Optimization and Hyperparameters We use SGD with Nesterov momentum 0.9 and a weight decay of 5×10^{-4}. For WRN, we apply a dropout $p = 0.1$ at train time. We perform extensive cross-validation of all the hyperparameters for all the baselines.

Hyperparameters For all our experiments we set the batch size to 128[1]. At training time, we apply standard augmentation (random cropping and horizontal flipping, similarly to [Liu et al., 2020]). The data is appropriately normalized before being fed to the network both at train and test time.

- for DNN-SN, DNN-SRN and Mix-MaxEnt the set of SN clamping factors we considered in our experiments are $c \in \{0.5, 0.75, 1.0\}$, the target of stable rank $r \in \{0.3, 0.5, 0.7, 0.9\}$ (as $r = 1$ for SRN is the same as applying SN with $c = 1.0$).
- for Mixup, we considered the Beta distribution hyperparameters to be $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, as suggested in the literature [Thulasidasan et al., 2019].
- for Mix-MaxEnt we considered the Beta distribution hyperparameters to be $\alpha \in \{5, 10, 15, 20, 30\}$. In both cases we set $\alpha = \beta$ in $\texttt{Beta}(\alpha, \beta)$.
- for the KFAC-LLLA, we took 1000 samples. Although the number might seem quite high, we could not notice significant improvements using a lower number of samples. We tuned the prior variance σ_0 needed for the computation of the Laplace approximation minimising the ECE on the validation set. We also tried using the theoretical value $\sigma_0 = 1/\tau$ [Kristiadi et al., 2020], where τ represents the weight decay, but it produced inferior results with respect to our cross-validation procedure.
- for Deep Ensembles we use 5 members.
- when temperature scaling is applied, the temperature T is tuned on the validation set, minimising the ECE (we considered values ranging from 0.1 to 10, with a step size of 0.01). For Deep Ensembles, we first compute the mean of the logits, then scale it by the temperature parameter before passing it through the softmax.

All the hyperparameters we chose are reported in Table 3. Such hyperparameters have been selected performing cross-validation with stratified-sampling on a 90/10 split of the training set to maximise Accuracy[2]. Indeed, it is important to observe that:

- optimising the hyperparameters based solely on the ECE can prefer models with lower Accuracy but better calibration. However, Accuracy is commonly considered of primary importance, and any method improving calibration should avoid degrading it.
- Optimising considering any of the corrupted experiments and OOD detection metrics would be equivalent to *overfitting the test set* (indeed, both corruptions and OOD datasets are assumed to be unknown at training time by all the considered methodologies, and so they should be during the hyperparameter selection procedure).

For all of our experiments we train the methods initialising the networks with 5 different seeds and report the average and standard deviation *in percentage* for all the metrics. For the hyperparameter search, we trained approximately 350 models. For the final tables (five seeds), we trained approximately 250 models.

Code For fair comparisons, we developed our own code base for all the approaches mentioned above (except SNGP and DUQ) and performed an extensive hyperparameter search to obtain the strongest possible baselines. For SNGP, we used the available code and made sure that we follow exactly the same procedure as mentioned in their original paper. For DUQ, the original paper did not perform large scale experiments similar to ours. Unfortunately, we could not manage to make their code work on C100 as it exhibited unstable behaviour. For this reason, we borrowed numbers for DUQ from the SNGP paper. Please note that the authors of SNGP performed non-trivial modifications to the original DUQ methodology to make it work on C100. For the SNGP method we use the official code-base with the suggested hyperparameters and training procedures. The code diverges slightly from the procedure described in their paper, hence the slight differences in the performance. The only

[1] For SNGP and DUQ, we use the hyperparameters suggested in their original papers.

[2] Except for the σ_0 of the KFAC-LLLA, as we could not observe significant differences in Accuracy between hyperparameters optimising the Accuracy and ECE

Training Set Architecture	Hyp	**C10** WRN	**C10** R50	**C100** WRN	**C100** R50
DNN	T	1.3	1.5	1.3	1.4
DNN-SN	c	0.5	0.5	0.5	0.5
	T	1.4	1.5	1.2	1.4
DNN-SR	r	0.3	0.3	0.3	0.3
	T	1.3	1.4	1.2	1.4
DE	T	1.3	1.4	1.1	1.2
SNGP	T	1.4	-	1.5	-
Mixup	α	0.3	0.3	0.3	0.3
	T	0.7	0.8	1.09	1.2
Mix-MaxEnt	α	20	20	10	10
	T	1.1	1.3	1.2	1.2
Mix-MaxEnt-SN	α	20	20	20	20
	T	1.23	1.23	1.09	1.19
	c	0.5	0.5	0.5	0.5
Mix-MaxEnt-SRN	α	20	30	20	10
	T	1.2	1.3	1.09	1.09
	r	0.9	0.9	0.9	0.9
KFAC-LLLA	samples	1000	1000	1000	1000
	σ_0	1	0.6	4	0.1

Table 3: Hyperparameters selected via cross-validation (90/10 split with stratified sampling) for all the methods we trained from scratch. We tuned T and σ_0 by minimising the ECE. All other hyperparameters have been tuned to maximise the Accuracy.

modification we performed to the official code-base was to make the inference procedure consistent with the one described in the paper: indeed, in their code they implement a mean-field approximation to estimate the predictive distribution [Lu et al., 2020], while in their paper they use Monte Carlo Integration with a number of samples equal to the number of members in the ensembles they use as a baseline, which provides better calibration. The rationale is that we could not find an obvious way to tune the mean-field approximation hyperparameters to improve at the same time both the calibration and OOD detection performance (indeed, the mean-field approximation imposes a trade-off between calibration and OOD detection performance). Additionally, since the standard KFAC-LLLA uses the same Monte Carlo Integration procedure, we opted for the latter for a fair comparison. For the KFAC-LLLA we leverage the official repository[3] [Hobbhahn et al., 2021] and the Backpack library [Dangel et al., 2020] for the computation of the Kronecker-Factored Hessian. For the SNGP ResNet50 experiments, we tried running the official implementation. The official implementation for ResNet50 is specifically fine-tuned for ImageNet, and has not been used for experiments on CIFAR. We could not make SNGP converge to SOTA accuracy values both on CIFAR-10 and CIFAR-100. All the other methods were implemented by us in PyTorch and the training, cross-validation and evaluation code will be made publicly available upon acceptance of the paper.

B Additional Results

B.1 ResNet50 Experiments

In Tables 4 and 5 we report the experimental results for ResNet50.

B.2 Calibration Metrics without Temperature Scaling

For completeness, we report the calibration metrics over all the methods and considered datasets without temperature scaling [Guo et al., 2017] in Table 6. We can observe that temperature scaling leaves the ranking among methods mostly unchanged. Details about the cross-validation procedure used and the temperature T used are in Section A.

[3] `https://github.com/19219181113/LB_for_BNNs`

Methods	Clean			Corrupted			CIFAR100		SVHN	
	Accuracy (↑)	**ECE** (↓)	**AdaECE** (↓)	**Accuracy** (↑)	**ECE** (↓)	**AdaECE** (↓)	AUROC (↑)	AUPR (↑)	AUROC (↑)	AUPR (↑)
DNN	95.19 ± 0.23	1.38 ± 0.19	1.45 ± 0.19	75.18 ± 0.69	12.31 ± 0.84	12.29 ± 0.85	88.61 ± 0.66	88.05 ± 0.52	93.20 ± 1.98	96.43 ± 0.95
DNN-SN	95.20 ± 0.15	1.11 ± 0.09	1.27 ± 0.10	74.88 ± 0.96	11.75 ± 0.48	11.74 ± 0.49	88.19 ± 0.36	87.72 ± 0.32	93.46 ± 3.41	96.56 ± 1.87
DNN-SRN	95.39 ± 0.20	1.23 ± 0.08	1.27 ± 0.13	75.40 ± 0.67	12.22 ± 0.64	12.20 ± 0.64	88.82 ± 0.40	88.15 ± 0.31	93.54 ± 2.41	96.63 ± 1.27
Deep Ensembles	96.23 ± 0.05	1.27 ± 0.05	1.28 ± 0.03	77.63 ± 0.36	13.12 ± 0.32	12.68 ± 0.32	$\mathbf{91.38 \pm 0.21}$	$\mathbf{90.75 \pm 0.13}$	96.90 ± 0.07	98.27 ± 0.09
KFAC-LLLA	95.21 ± 0.26	0.79 ± 0.26	$\mathbf{0.69 \pm 0.24}$	75.18 ± 0.89	10.26 ± 0.97	$\mathbf{10.23 \pm 0.97}$	89.54 ± 0.41	88.30 ± 0.41	93.13 ± 1.01	96.25 ± 0.63
Mixup	96.05 ± 0.15	$\mathbf{0.59 \pm 0.39}$	2.17 ± 0.51	78.63 ± 0.72	$\mathbf{10.17 \pm 0.91}$	10.35 ± 0.95	84.24 ± 2.95	85.35 ± 1.76	89.40 ± 4.35	95.57 ± 1.41
Mix-MaxEnt	$\mathbf{96.69 \pm 0.17}$	0.65 ± 0.11	0.94 ± 0.21	$\mathbf{81.16 \pm 1.48}$	12.61 ± 1.77	12.52 ± 1.78	87.63 ± 0.67	85.85 ± 0.83	$\mathbf{94.39 \pm 0.72}$	$\mathbf{96.31 \pm 0.49}$

Table 4: ResNet50 trained on C10. The cross-validated hyperparameters are provided in Appendix A.

Methods	Clean			Corrupted			CIFAR10		SVHN	
	Accuracy (↑)	**ECE** (↓)	**AdaECE** (↓)	**Accuracy** (↑)	**ECE** (↓)	**AdaECE** (↓)	AUPR (↑)	AUROC (↑)	AUPR (↑)	AUROC (↑)
DNN	79.19 ± 0.44	3.05 ± 0.29	2.94 ± 0.31	50.62 ± 0.42	19.80 ± 0.31	19.76 ± 0.31	79.33 ± 0.70	75.20 ± 0.61	82.45 ± 3.21	89.92 ± 1.99
DNN-SN	79.27 ± 0.25	3.15 ± 0.12	3.13 ± 0.15	50.55 ± 0.39	12.19 ± 0.47	12.16 ± 0.47	79.20 ± 0.17	75.22 ± 0.13	80.78 ± 1.08	88.87 ± 0.84
DNN-SRN	78.96 ± 0.42	2.98 ± 0.24	2.95 ± 0.24	50.48 ± 0.37	12.62 ± 0.58	12.59 ± 0.58	78.77 ± 0.14	74.87 ± 0.15	82.39 ± 2.83	89.52 ± 1.75
Deep Ensembles	$\mathbf{82.09 \pm 0.33}$	3.15 ± 0.10	2.98 ± 0.19	53.91 ± 0.37	12.53 ± 0.31	12.36 ± 0.31	$\mathbf{81.93 \pm 0.28}$	$\mathbf{77.65 \pm 0.34}$	85.08 ± 1.60	91.49 ± 0.88
KFAC-LLLA	79.41 ± 0.44	$\mathbf{1.30 \pm 0.09}$	$\mathbf{1.19 \pm 0.24}$	50.85 ± 0.49	$\mathbf{10.59 \pm 0.56}$	$\mathbf{10.57 \pm 0.56}$	79.30 ± 0.41	75.27 ± 0.38	82.80 ± 3.84	90.38 ± 2.17
Mixup	80.12 ± 0.28	7.49 ± 0.32	7.47 ± 0.35	53.96 ± 0.21	13.57 ± 0.38	13.52 ± 0.38	77.02 ± 0.41	74.40 ± 0.43	76.86 ± 3.40	87.36 ± 1.62
Mix-MaxEnt	81.49 ± 0.31	1.57 ± 0.18	1.53 ± 0.21	$\mathbf{57.62 \pm 0.30}$	13.42 ± 0.93	13.39 ± 0.94	79.44 ± 0.33	75.80 ± 0.14	$\mathbf{88.68 \pm 0.69}$	$\mathbf{93.48 \pm 0.34}$

Table 5: ResNet50 trained on C100. The cross-validated hyperparameters are provided in Appendix A.

Methods	Clean		Corrupted	
	ECE (↓)	**AdaECE** (↓)	**ECE** (↓)	**AdaECE** (↓)
C10 R50				
DNN	3.06 ± 0.20	3.02 ± 0.19	17.31 ± 0.73	17.30 ± 0.73
DNN-SN	2.92 ± 0.11	2.90 ± 0.11	17.41 ± 1.02	17.40 ± 1.02
DNN-SRN	2.85 ± 0.16	2.82 ± 0.16	17.18 ± 0.62	17.17 ± 0.62
Deep Ensembles	2.12 ± 0.01	2.10 ± 0.03	14.01 ± 0.32	13.99 ± 0.32
KFAC-LLLA	0.84 ± 0.26	0.76 ± 0.24	11.59 ± 0.62	11.52 ± 0.62
Mixup	2.84 ± 0.54	2.87 ± 0.46	11.17 ± 0.91	11.35 ± 0.95
Mix-MaxEnt (**Ours**)	1.64 ± 0.15	1.42 ± 0.13	12.21 ± 1.84	12.12 ± 1.84
C100 R50				
DNN	9.51 ± 0.58	9.47 ± 0.60	25.18 ± 1.46	25.17 ± 1.47
DNN-SN	9.47 ± 0.44	9.44 ± 0.46	25.06 ± 0.73	25.04 ± 0.73
DNN-SRN	9.63 ± 0.23	9.59 ± 0.23	25.51 ± 0.69	25.50 ± 0.69
Deep Ensembles	6.65 ± 0.12	6.50 ± 0.04	19.80 ± 0.31	19.76 ± 0.31
KFAC-LLLA	1.56 ± 0.09	1.49 ± 0.24	12.11 ± 1.12	12.18 ± 1.12
Mixup	7.49 ± 0.32	7.47 ± 0.35	21.53 ± 0.38	21.52 ± 0.38
Mix-MaxEnt (**Ours**)	4.33 ± 0.43	4.17 ± 0.39	14.06 ± 1.81	14.03 ± 1.80
C10 WRN				
DNN	2.30 ± 0.11	2.27 ± 0.11	15.94 ± 0.66	15.92 ± 0.66
DNN-SN	2.25 ± 0.12	2.21 ± 0.13	15.55 ± 0.23	15.53 ± 0.23
DNN-SRN	2.25 ± 0.12	2.23 ± 0.13	15.13 ± 0.44	15.11 ± 0.44
Deep Ensembles	1.76 ± 0.02	1.74 ± 0.03	13.54 ± 0.19	13.52 ± 0.18
SNGPGood	1.62 ± 0.09	1.51 ± 0.06	11.36 ± 0.37	11.33 ± 0.36
KFAC-LLLA	1.06 ± 0.08	1.12 ± 0.07	11.69 ± 0.76	11.67 ± 0.76
Mixup	2.02 ± 0.72	2.23 ± 0.64	7.88 ± 0.94	7.93 ± 0.97
Mix-MaxEnt (**Ours**)	0.92 ± 0.06	0.71 ± 0.13	8.93 ± 0.70	8.87 ± 0.69
C100 WRN				
DNN	5.34 ± 0.38	5.30 ± 0.42	17.43 ± 0.75	17.38 ± 0.75
DNN-SN	5.15 ± 0.25	4.97 ± 0.24	16.39 ± 0.44	16.35 ± 0.45
DNN-SRN	5.12 ± 0.17	5.05 ± 0.25	15.75 ± 0.85	15.71 ± 0.85
Deep Ensembles	4.03 ± 0.16	3.92 ± 0.16	13.51 ± 0.28	13.47 ± 0.29
SNGPGood	5.68 ± 0.26	5.65 ± 0.28	10.97 ± 1.52	10.89 ± 1.52
KFAC-LLLA	2.20 ± 0.31	2.30 ± 0.32	8.97 ± 0.21	8.99 ± 0.21
Mixup	3.44 ± 1.08	3.60 ± 0.99	16.53 ± 1.52	16.54 ± 1.49
Mix-MaxEnt (**Ours**)	2.66 ± 0.07	2.47 ± 0.09	10.54 ± 1.49	10.49 ± 1.49

Table 6: Calibration metrics for all networks and all datasets, without temperature scaling

C Detailed analysis of the corruption experiments

In this section, we report detailed plots that break down the aggregated metric reported in Table 2 for CIFAR-10-C and WideResNet28-10. Specifically, in Figures 2, 3 and 4 we show how the Accuracy, ECE and AdaECE vary across all the corruption types and intensity values (horizontal axis) over 5 seeds. As it can be seen, in most cases our method achieves better Accuracy than any other method. In terms of calibration, our method is sometimes outperformed by Mixup, but we observe that Mixup exhibits lower Accuracy whenever this happens. As the intensity of the corruption increases, all the metrics deteriorate for all the methods (as expected). However, our method still achieves superior performance in most of the cases also in these circumstances.

To support our claim that better clustering behaviour induces better classification performance, we report the Fisher criterion, $||S_W||_F$ and $||S_B||_F$ (we defined in Section 4) plots for all corruptions and all intensity levels in Figures 5, 6 and 7.

The observed pattern is similar for all considered architectures and corrupted datasets, hence we report these plots only for WideResNet28-10 on CIFAR-10-C.

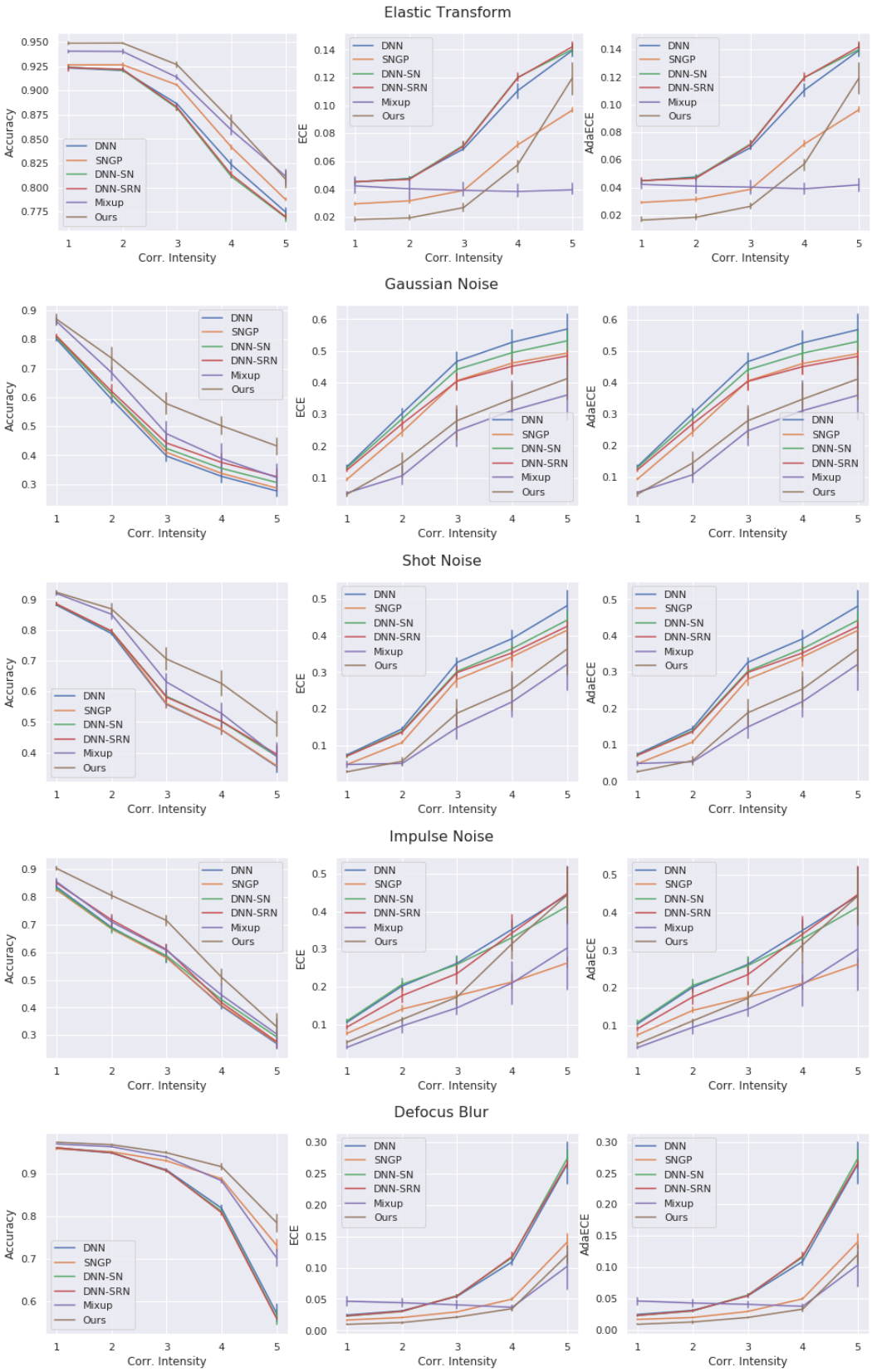

Figure 2: (Part 1 of 3) Accuracy, ECE and AdaECE for all corruptions and intensity values of CIFAR-10-C, architecture WideResNet28-10. A similar pattern can be observed in all other cases.

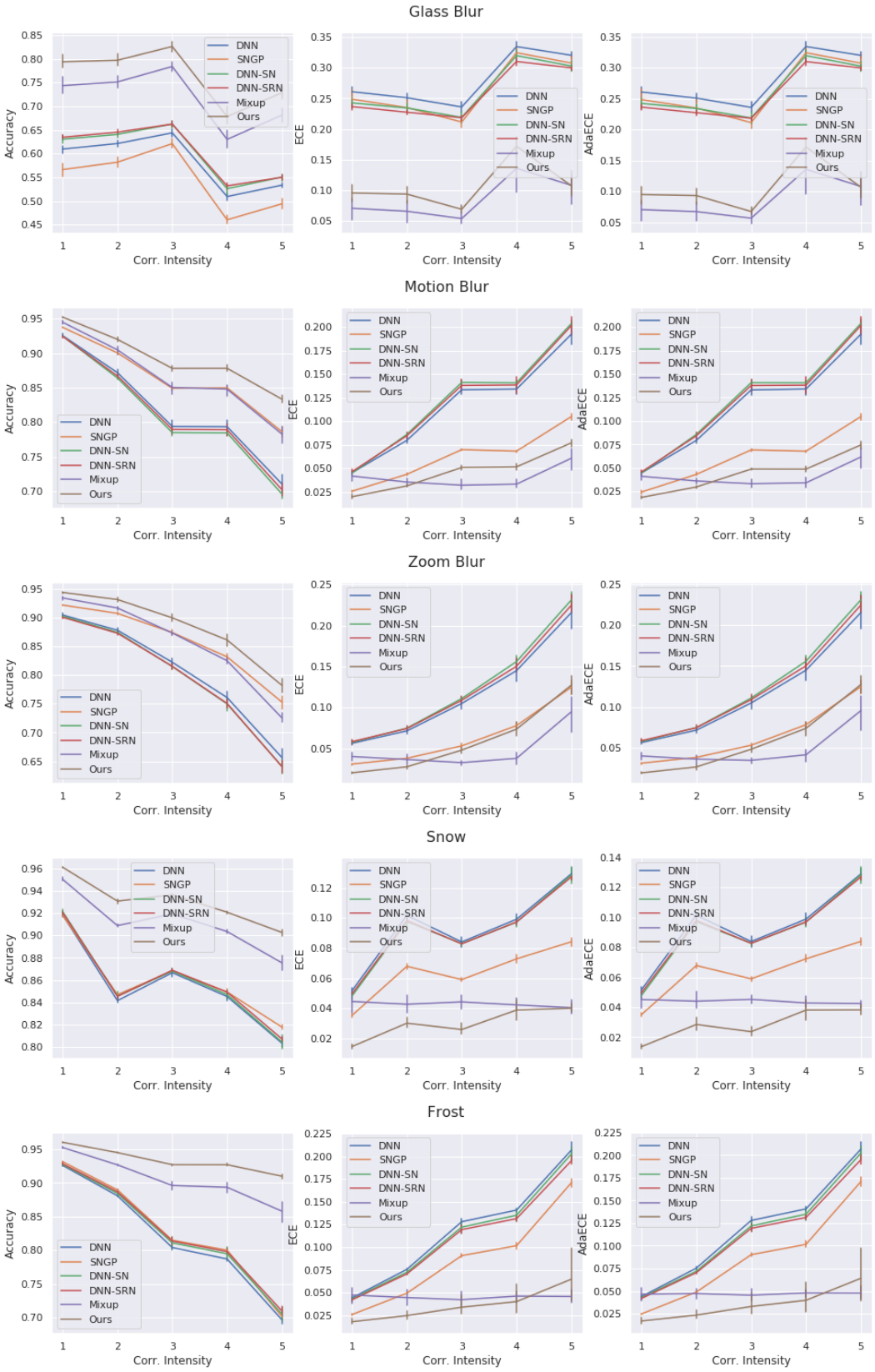

Figure 3: (Part 2 of 3) Accuracy, ECE and AdaECE for all corruptions and intensity values of CIFAR-10-C, architecture WideResNet28-10.

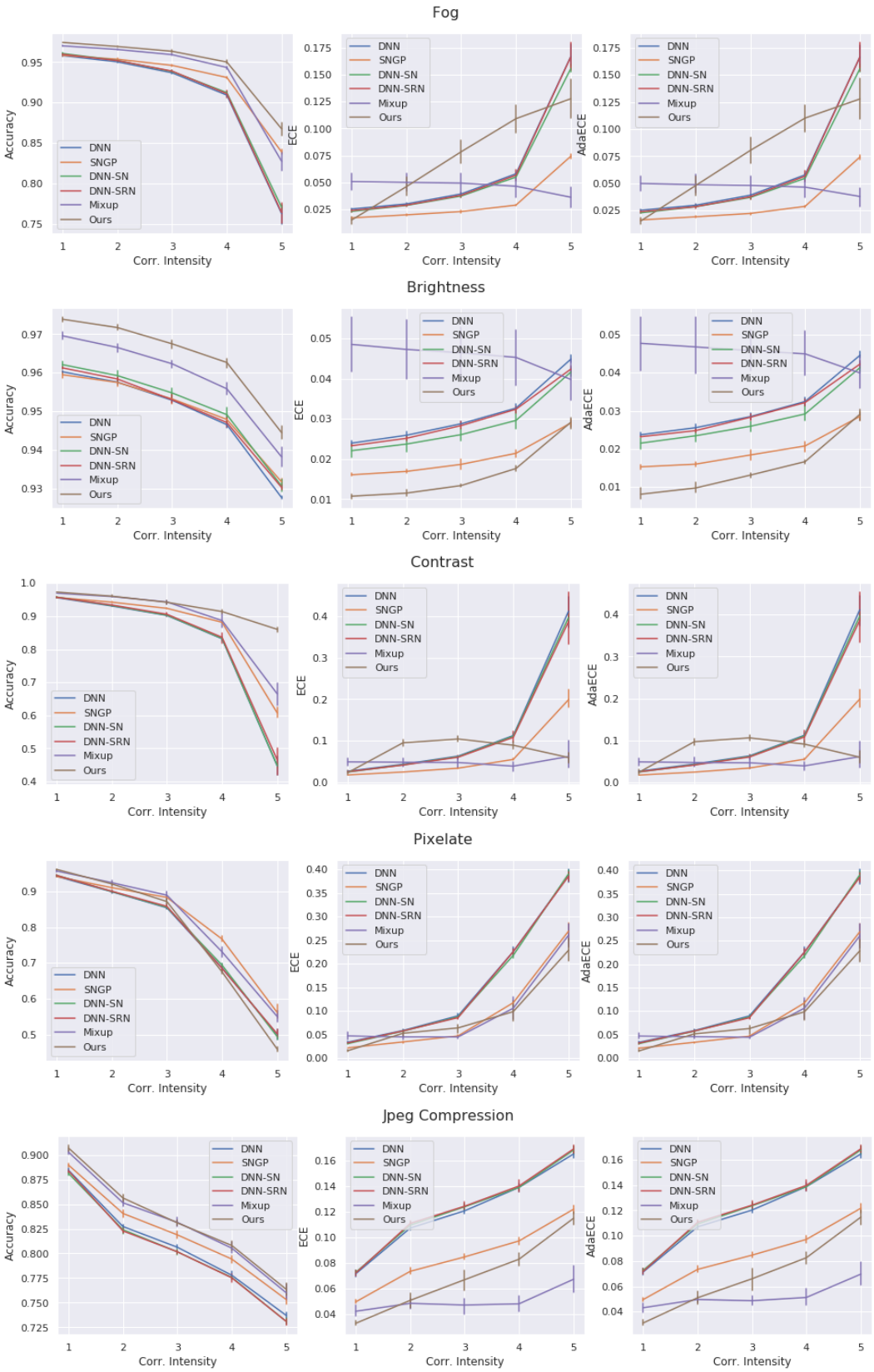

Figure 4: (Part 3 of 3) Accuracy, ECE and AdaECE for all corruptions and intensity values of CIFAR-10-C, architecture WideResNet28-10.

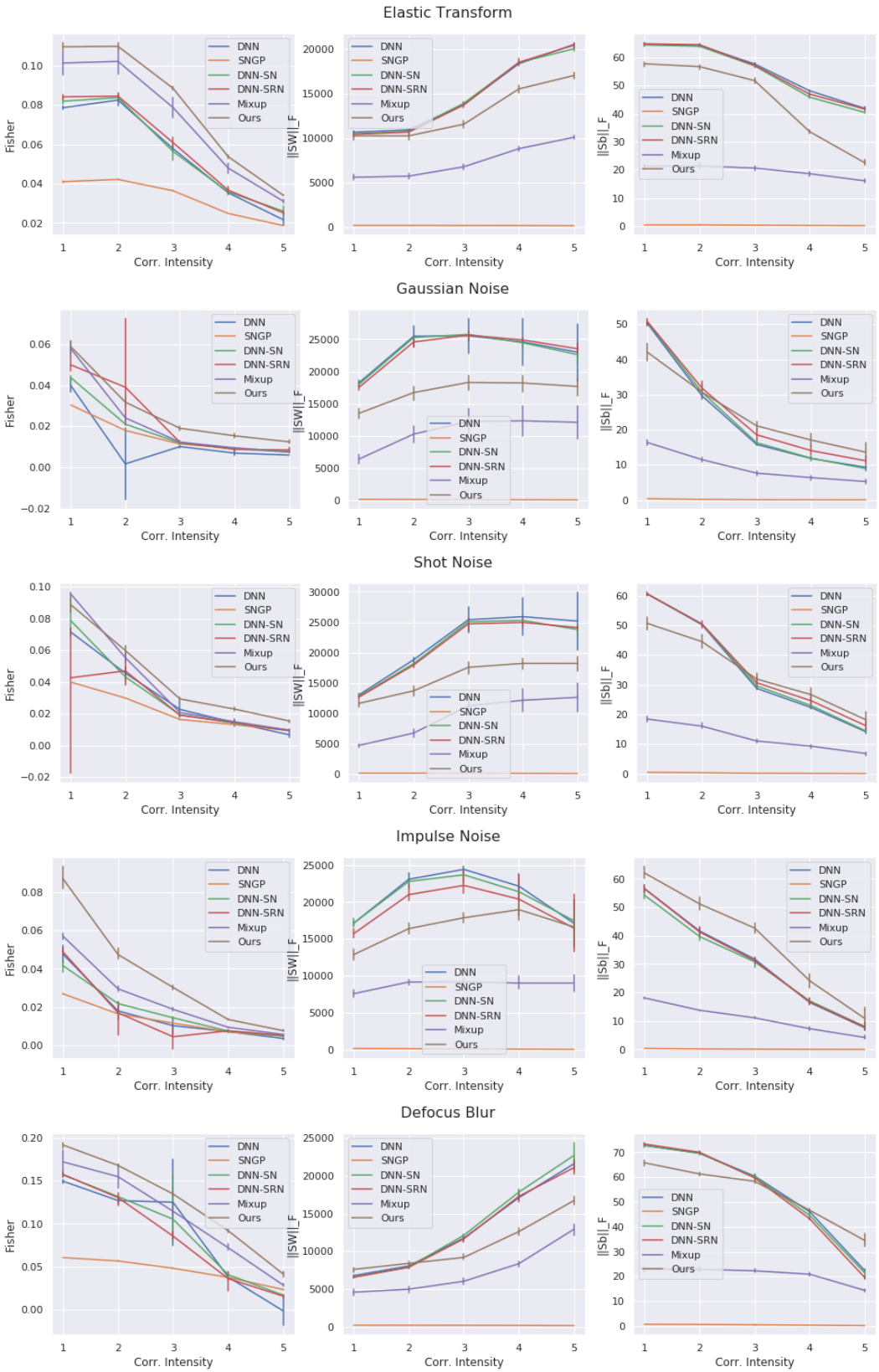

Figure 5: (Part 1 of 3) Fisher criterion, $||S_W||_F$ and $||S_B||_F$ for all corruptions and intensity values of CIFAR-10-C, architecture WideResNet28-10. A similar pattern can be observed in all other cases.

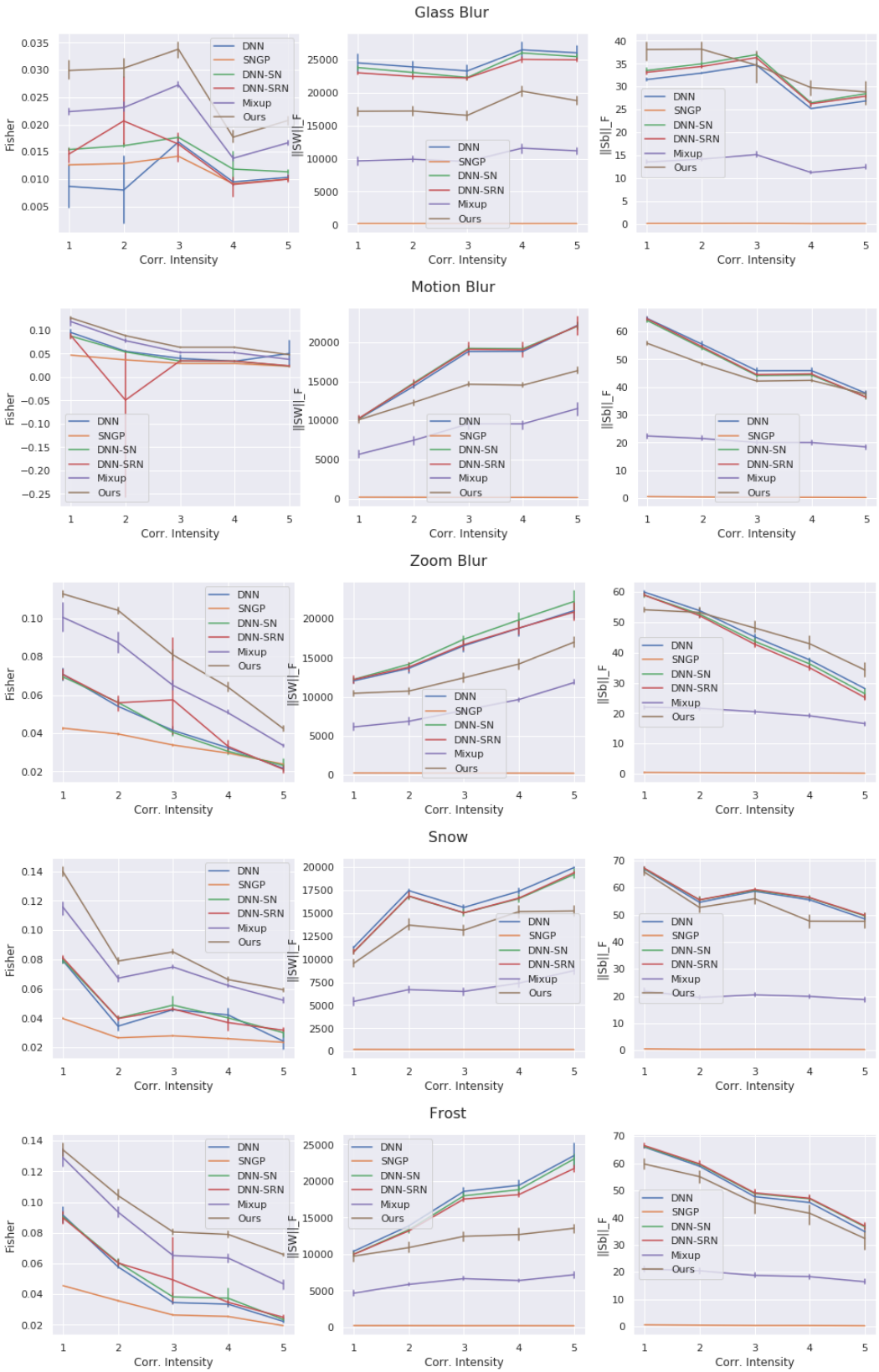

Figure 6: (Part 2 of 3) Fisher criterion, $||S_W||_F$ and $||S_B||_F$ for all corruptions and intensity values of CIFAR-10-C, architecture WideResNet28-10.

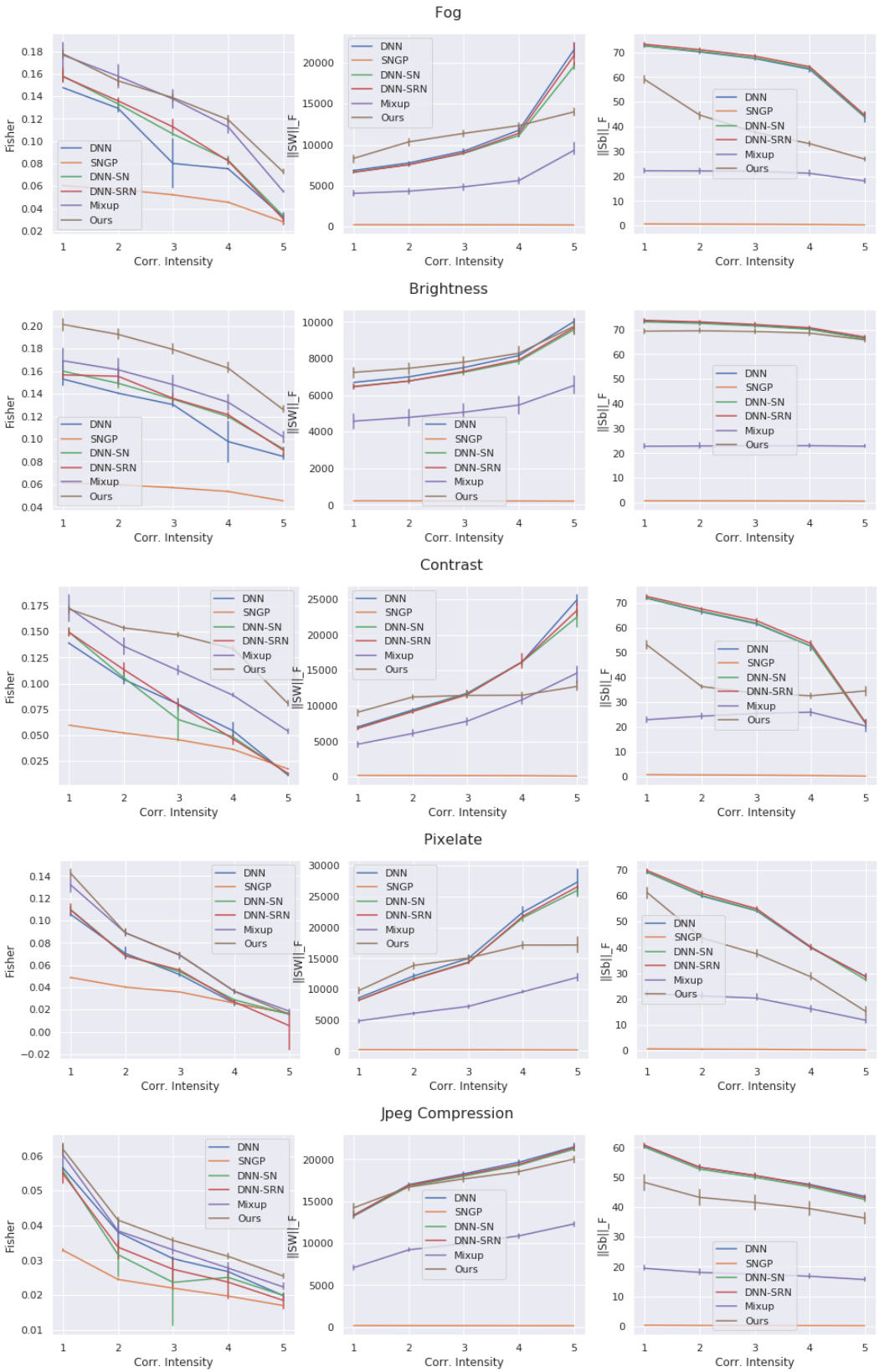

Figure 7: (Part 3 of 3) Fisher criterion, $||S_W||_F$ and $||S_B||_F$ for all corruptions and intensity values of CIFAR-10-C, architecture WideResNet28-10.