Diverse Ensembles Improve Calibration

Asa Cooper Stickland [1] **Iain Murray** [1]

Abstract

Modern deep neural networks can produce badly calibrated predictions, especially when train and test distributions are mismatched. Training an ensemble of models and averaging their predictions can help alleviate these issues. We propose a simple technique to improve calibration, using a different data augmentation for each ensemble member. We additionally use the idea of 'mixing' un-augmented and augmented inputs to improve calibration when test and training distributions are the same. These simple techniques improve calibration and accuracy over strong baselines on the CIFAR10 and CIFAR100 benchmarks, and out-of-domain data from their corrupted versions.

1. Introduction

Modern neural network models can produce overconfident or miscalibrated predictions, even when training examples are independent and identically distributed (i.i.d.) to the test distribution. This miscalibration is exacerbated when the training and testing distributions are different. In safety-critical scenarios, the ability to accurately represent model uncertainty is valuable.

Such model miscalibration has been shown to reduce when we train an ensemble of models and average their predictions (Lakshminarayanan et al., 2017; Ovadia et al., 2019). Ensembles have long been known to improve generalisation (Hansen & Salamon, 1990), especially when an ensemble is diverse, which is promoted with various techniques such as using latent variables (Sinha et al., 2020) or diversity-encouraging losses and architecture changes (Kim et al., 2018; Lee et al., 2016; Pang et al., 2019). Recent work proposes 'cheap' ensembles by sharing most of the model parameters across all ensembles, and using rank-1 factors to modify the linear layers in an ensemble member (Wen et al., 2020), making ensembles easier to train and store.

[1]School of Informatics, University of Edinburgh. Correspondence to: Asa Cooper Stickland <a.cooper.stickland@ed.ac.uk>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

Another long-standing way to improve generalization is *Data Augmentation*, i.e. expanding our training set with modified copies (Zhang et al., 2018; Yun et al., 2019; Cubuk et al., 2019). Recent examples include work by Hendrycks et al. (2020) and Xie et al. (2019). These approaches exploit the intuition that a blurry or rotated image should have the same class as the original image.

This work extends and combines recent work on cheap ensembles and data augmentation. We increase ensemble diversity by applying different augmentations to each ensemble member. This method improves calibration on an i.i.d. test set, and accuracy and calibration on out-of-distribution test sets for the CIFAR10 and CIFAR100 datasets. We additionally simplify the idea of 'mixing' un-augmented and augmented inputs introduced by Hendrycks et al. (2020), and explore adversarial perturbations, which apply more generally and result in better performance on i.i.d. data.

2. Methods

The computational and space costs of training independent ensembles scale linearly with the number of ensembles, so we share parameters as in BatchEnsemble (Wen et al., 2020). Each weight matrix in ensemble member i is a Hadamard product $W_i = W \circ r_i s_i^\top$, where W is shared across all ensemble members, and r_i and s_i are vectors that adapt the weights for this member. The adaptation can be implemented efficiently by elementwise multiplication of hidden states by s_i before multiplication by W, and r_i after.

We take a batch of B training examples, repeat it K times, where K is the number of ensemble members, and apply different augmentations to each copy of the original batch. When we pass the examples through the network, each copy of an example sees a different set of adapted weights.

We aim to test if different augmentations in each ensemble member gives better calibration, we use three augmentations: 1) *Adversarial perturbations:* Images are perturbed to increase training loss, adapting the 'fast gradient sign method' (Goodfellow et al., 2015). 2) AUGMIX: Augmentations from Hendrycks et al. (2020), with minor modifications. 3) *Stochastic Depth* (Huang et al., 2016; Fan et al., 2020). We randomly drop residual connections. With the exception of Stochastic Depth, we can apply these augmen-

tations to the input data and require no modification of the neural network architecture.

2.1. Adversarial Perturbations

We generate a new input image x_{adv}, from the original {image, label} pair $\{x, y\}$ as follows:

$$x_{\text{adv}} = x + \frac{m}{\mathbb{E}(m)} \cdot u \cdot s \cdot \text{sign}(\nabla L(x, y)), \qquad (1)$$

where $L(x, y)$ is the training loss, $u \sim U(0, 1)$, $m \sim \text{Bernoulli(p)}$ and s is a constant, the 'severity' of augmentation that varies per ensemble member. Without u and m, it's the *fast gradient sign method* (Goodfellow et al., 2015).

We introduced u to create a distribution of perturbations, and m so that we sometimes get an unperturbed image as input. Preliminary experiments suggested that these additional terms improved performance. The random m and u are scalars, i.e. they vary by input example but not by input dimension. The m term, scaled by its expectation, is taken from Dropout (Srivastava et al., 2014).

2.2. AUGMIX

The AUGMIX method (Hendrycks et al., 2020) aims to make models robust to out-of-distribution data by exposing the model to a wide variety of augmented images. The augmentation operations are from AutoAugment (Cubuk et al., 2019), excluding the operations used to create out-of-domain corrupted test sets (Section 3.1). Most operations have a varying 'severity', e.g. rotation by 2° or 15°. Several augmentation 'chains' are sampled, where a chain is a composition of one to three randomly selected operations. The augmented images from each chain are combined with a random convex combination, see the 'Augment' function in Algorithm 1.

The final stage of AUGMIX combines the original and augmented image with a convex combination sampled from $\text{Beta}(\alpha, \alpha)$. Initial results of Hendrycks et al. (2020) suggest a bimodal ($\alpha = 0.1$) Beta distribution performed best — sometimes using an image close to the original one. We consider a simpler approach: simply picking the original or augmented image using a Bernoulli trial, as we did in Section 2.1. To encourage diversity, we additionally use a different augmentation severity per ensemble member. Our modified AUGMIX procedure is described in Algorithm 1.

The full AUGMIX method encourages consistency across predictions for diverse augmentations of the same input, through the use of the Jensen–Shannon divergence as a consistency loss. Note this consistency loss can be applied to one model, and could be used for each ensemble member separately (i.e. it is not intended to ensure members agree). However due to the extra complication of using the consistency loss, and the doubling or tripling of the batch size to generate multiple samples of the same batch (on top of increasing the batch size for the BatchEnsemble method), we did not use this aspect of the AUGMIX method.

Algorithm 1 Modified AUGMIX Pseudocode

Input: Image x_{orig}, Severity Vector $\mathbf{s}$, Ensemble Index i
function Augment(x_{orig}, s, $k=3$, $\alpha=1$)
  Fill x_{aug} with zeros
  Sample mixing weights:
    $(w_1, w_2, \ldots, w_k) \sim \text{Dirichlet}(\alpha, \alpha, \ldots, \alpha)$
  for $i = 1, \ldots, k$ **do**
    Sample aug. operations $\text{op}_1, \text{op}_2, \text{op}_3 \sim \mathcal{O}$
    Compose operations with varying depth (with severity s)
    $\text{op}_{12} = \text{op}_2 \circ \text{op}_1$ and $\text{op}_{123} = \text{op}_3 \circ \text{op}_2 \circ \text{op}_1$
    Sample uniformly chain $\sim \{\text{op}_1, \text{op}_{12}, \text{op}_{123}\}$
    $x_{\text{aug}} \mathrel{+}= w_i \cdot \text{chain}(x_{\text{orig}})$
  end for
  return x_{aug}
end function
function AugmentAndMix(x_{orig}, s, $p=0.875, \beta=1.0$)
  $x_{\text{aug}} = \text{Augment}(x_{\text{orig}}, s)$
  Sample weight $m \sim$ Bernoulli(p) or Beta(β, β)
  Interpolate $x_{\text{augmix}} = (1 - m)x_{\text{orig}} + m x_{\text{aug}}$
  return x_{augmix}
end function
$x^i_{\text{augmix}} = \text{AugmentAndMix}(x_{\text{orig}}, s=\mathbf{s}_i)$

3. Experimental Settings

3.1. Datasets

We use the CIFAR-10 and CIFAR-100 classification datasets of tiny natural images (Krizhevsky, 2009). For each task, we hold out a validation set of 5,000 random images from the training set, and train the best-performing models on the entire training set and evaluate on the test set. To test robustness to out-of-distribution data, we evaluate on corrupted versions of the test sets, CIFAR-10-C and CIFAR-100-C (Hendrycks & Dietterich, 2019). There are 15 noise, blur, weather, and digital corruption types, each appearing at 5 severity levels or intensities. We do not use these corruptions in training.

3.2. Metrics

When a well calibrated model is, say, 60% confident, it will be correct 60% of the time. A measure of this is *Expected Calibration Error* (ECE; Guo et al., 2017). We divide the data into m equally sized bins B_m and measures the absolute difference between average confidence and accuracy in each bin, i.e. $\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|$, where n is the number of examples. An alternative (Hendrycks et al., 2020) uses squared difference instead of absolute, and takes the square root of the sum, which we call 'ECE-rms'. We always use 15 bins. For corrupted data we sum error over the corruption intensities, and average over the 15 corruption types (Hendrycks et al., 2020).

Table 1. Comparing performance when using the AUGMIX (referred to as 'AM' in the table) method, with either a Beta distribution to mix augmented images, or a Bernoulli ('Bern.') distribution, with p being the probability of augmenting an input image. 'No B.E.' refers to using a single model, and otherwise we use BatchEnsemble. We highlight in bold the best result for each metric, separately for the test set.

METHOD	CIFAR 10 VAL. ERR.	VAL. ECE	VAL. ECE-RMS	CIFAR 100 VAL. ERR.	VAL. ECE	VAL. ECE-RMS
AM BETA	3.82	1.41	3.60	**18.5**	4.69	5.6
AM BETA (NOT DIVERSE)	3.88	1.48	3.19	**18.5**	4.76	**5.46**
AM BERN. ($p = 0.875$)	3.46	1.26	2.63	18.6	**4.47**	5.58
AM BERN. ($p = 1.0$)	3.64	1.38	3.28	18.8	5.07	5.90
AM BERN. ($p = 0.875$) + ADV.	**3.44**	**1.15**	**1.99**	19.0	5.21	6.11
	TEST			TEST		
	ERR.	ECE	ECE-RMS	ERR.	ECE	ECE-RMS
AM (NO B.E.) BERN. ($p = 0.875$)	3.98	1.15	2.13	21.0	7.41	9.37
AM BERN. ($p = 0.875$)	3.40	1.15	2.04	17.7	4.95	6.08
AM BERN. ($p = 0.875$) + ADV.	**3.13**	**1.00**	**1.88**	**17.6**	**4.51**	**5.36**
	CIFAR 10-C			CIFAR 100-C		
	ERR.	ECE	ECE-RMS	ERR.	ECE	ECE-RMS
AM BETA	11.6	**4.30**	**5.61**	34.0	10.2	11.0
AM BETA (NOT DIVERSE)	11.4	4.39	5.88	34.3	10.7	11.7
AM BERN. ($p = 0.875$)	11.5	4.78	6.147	34.0	**9.92**	**10.8**
AM BERN. ($p = 1.0$)	**11.3**	4.63	5.99	**33.7**	10.7	11.7
AM BERN. ($p = 0.875$) + ADV.	11.6	4.72	6.07	34.2	10.8	11.8
	TEST			TEST		
	ERR.	ECE	ECE-RMS	ERR.	ECE	ECE-RMS
AM (NO B.E.) BERN. ($p = 0.875$)	12.8	5.27	6.93	35.8	14.8	16.6
AM BERN. ($p = 0.875$)	11.0	4.32	5.71	**33.2**	10.4	11.5
AM BERN. ($p = 0.875$) + ADV.	**10.6**	**4.08**	**5.40**	**33.2**	**10.2**	**11.2**

3.3. Training Setup

We use the ResNeXt-29 (32×4) architecture (Xie et al., 2017), which was the best-performing of those tested by Hendrycks et al. (2020). We use SGD with Nesterov momentum, an initial learning rate of 0.1 decayed following a cosine schedule (Loshchilov & Hutter, 2017), and weight decay of 0.0005. Input images are pre-processed with standard random left-right flipping and cropping prior to any augmentations. We train ensembles for 250 epochs, and otherwise train for 200 epochs, following Wen et al. (2020). We initialise the per-ensemble parameters (the vectors that produce the rank-1 modification of the weight matrices) for BatchEnsemble with a $N(1, 0.5^2)$ distribution, and always use 4 ensemble members. We use a batch size of 128, which is $4\times$ larger when using BatchEnsemble (each ensemble member sees a copy of the same 128 images).

We introduce a 'severity vector' $\mathbf{s}$, where each element of the vector corresponds to the severity of augmentation for a particular ensemble member. By varying this vector we can control whether each ensemble member gets the same input distribution, or a different one per ensemble member. Our augmentation hyperparameters are as follows:

- *Adversarial perturbations:* $\mathbf{s} = [0.0, 0.05, 0.1, 0.15]$, with s_i corresponding to s in eq 1. The probability of not using an augmented image was $p = 0.875$. When not using different severity per ensemble member, we randomly shuffle the elements of $\mathbf{s}$ before each update.
- AUGMIX: $\mathbf{s} = [1, 2, 3, 4]$ (severity in AUGMIX takes on integer values). When not using different severity per ensemble member, we set all elements to 3, the default value used by Hendrycks et al. (2020).
- *Stochastic Depth:* $\mathbf{s} = [0.0, 0.05, 0.1, 0.15]$, with the elements corresponding to probability of dropping a residual connection. When not using different severity per ensemble member, we set all elements to 0.075.

4. Experiments and Discussion

We group our results by those with the AUGMIX method (Table 1) and those without it (Table 2). Baselines (without augmentation) with single models and BatchEnsemble are at the top of Table 2.

4.1. Diverse Inputs vs. Not Diverse Inputs

For each of the augmentation types we considered, AUGMIX, adversarial perturbations and stochastic depth, using a different 'severity' of augmentation for each ensemble member tended to improve calibration on both i.i.d. and

Table 2. Accuracy and calibration for various methods on the cifar10 and cifar100 validation set (I.I.D. to the training distribution). 'B.E.' refers to 'BatchEnsemble'. '$p = 1.0$' refers to always using the perturbed input, rather than skipping with some probability as in eq 1. 'S.D.' refers to stochastic depth (randomly dropping residual connections). We highlight in bold the best result for each metric.

	CIFAR 10			CIFAR 100		
METHOD	VAL. ERR.	VAL. ECE	VAL. ECE-RMS	VAL. ERR.	VAL. ECE	VAL. ECE-RMS
VANILLA RESNEXT	4.18	**1.25**	3.20	21.0	6.88	8.06
+ BATCHENSEMBLE	3.96	1.57	3.07	19.4	5.04	5.81
B.E. + S.D.	**3.30**	1.45	2.76	19.6	8.50	10.2
B.E. + S.D. (NOT DIVERSE)	3.60	1.70	3.85	19.9	9.91	11.7
B.E. + ADVERSARY	3.60	1.29	**2.36**	19.5	4.75	5.57
B.E. + ADVERSARY (NOT DIVERSE)	3.46	1.34	2.64	**19.1**	**4.49**	**5.31**
B.E. + ADVERSARY ($p = 1.0$)	4.12	1.56	3.58	19.5	4.92	6.13

	CIFAR 10-C			CIFAR 100-C		
	ERR.	ECE	ECE-RMS	ERR.	ECE	ECE-RMS
VANILLA RESNEXT	26.5	13.5	15.4	50.1	19.9	22.1
+ BATCHENSEMBLE	27.0	**13.2**	**15.0**	50.8	19.9	21.1
B.E. + S.D.	26.3	14.2	16.2	**50.0**	24.7	26.2
B.E. + S.D. (NOT DIVERSE)	**26.0**	15.6	17.6	50.2	28.7	30.3
B.E. + ADVERSARY	26.2	**13.2**	**15.0**	50.6	**18.9**	**19.9**
B.E. + ADVERSARY (NOT DIVERSE)	26.5	14.3	16.1	50.8	19.9	21.0
B.E. + ADVERSARY ($p = 1.0$)	26.6	13.3	16.1	50.6	20.3	21.5

out-of-domain held-out data. Comparing methods to their 'not diverse' counterpart in Table 1 and Table 2, the ECE and ECE-rms scores for 'not diverse' ensembles are generally lower. This effect did not always hold for i.i.d. held out data, but it was always true for CIFAR10-C and CIFAR100-C. The error rate, however, showed no clear trend between the two types of ensemble.

Dropping residual connections ('stochastic depth') results in slightly better performance on CIFAR10, but decreases calibration performance on CIFAR10-C and CIFAR100-C.

4.2. The Importance Of Un-augmented Inputs

For the AUGMIX and adversarial perturbation methods, we compared two settings: augmenting every input image ($p = 1.0$ in Tables 1 and 2), or randomly mixing in some un-augmented images with 12.5% probability ($p = 0.875$ in Tables 1 and 2).

We find (Table 1 and Table 2), when using ensembles, that augmenting all inputs (with either AUGMIX or an adversary) results in worse performance on i.i.d. data, in terms of accuracy and calibration. For corrupted data the picture is mixed: for adversarial perturbation, augmenting all inputs gives similar or worse calibration, but for AUGMIX accuracy improves on corrupted images when augmenting all images (and calibration for CIFAR10-C). We also compare, in Appendix A, augmenting every input image and mixing in un-augmented images for single models (i.e. not BatchEnsemble) for AUGMIX (Table 3), however no clear picture emerges, although we note calibration improves for CIFAR100-C when augmenting every image.

For AUGMIX we additionally compare mixing augmented and un-augmented images with either a Bernoulli (Bern.) or Beta distribution (see Section 2.2). We found the simple baseline of 'no mixing' ($p = 1.0$ in Table 1) often outperformed the other methods, especially in corrupted error rate where it always performed best, but even on i.i.d. data improves over a Beta distribution for CIFAR-10. However adding a small probability of using un-augmented data ($p = 0.875$ in Table 1) improved calibration on i.i.d. data, which matches the intuition that exposing the model to some entirely in-distribution images at training time will help calibrate the model on i.i.d. held-out data.

Finally, we combine adversarial perturbations and AUGMIX in the same model. This combination performed best on CIFAR10. When comparing on the test set, using adversarial perturbations performs the best on all metrics. We also outperform the best models (when using AUGMIX with the Jensen-Shannon divergence described in Section 2.2) of Hendrycks et al. (2020), who achieve 10.9% error and 34.9% error on CIFAR10-C and CIFAR100-C respectively.

5. Conclusion

We present a simple method for increasing calibration in ensembles: using a different data augmentation for each ensemble member. We additionally avoid problems with miscalibration on i.i.d. data by randomly mixing in some un-augmented images. Overall our methods improve both calibration and accuracy across i.i.d. and out-of-distribution data. In future we hope to add theoretical understanding to our simple principles, perhaps automatically determining which augmentation to use. AUGMIX is well suited to image data, and we hope to expand to other domains and datasets, especially those with discrete inputs where the perturbations described here do not obviously apply.

Acknowledgements

We thank Artur Bekasov, Conor Durkan and James Ritchie for valuable discussion. Asa Cooper Stickland was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Fan, A., Grave, E., and Joulin, A. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Hansen, L. K. and Salamon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

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

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *ECCV*, 2016.

Kim, W., Goyal, B., Chawla, K., Lee, J., and Kwon, K. Attention-based ensemble for deep metric learning. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images, 2009. http://www.cs.toronto.edu/~kriz/cifar.html.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. Curran Associates, Inc., 2017.

Lee, S., Purushwalkam Shiva Prakash, S., Cogswell, M., Ranjan, V., Crandall, D., and Batra, D. Stochastic multiple choice learning for training diverse deep ensembles. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2119–2127. Curran Associates, Inc., 2016.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 13991–14002. Curran Associates, Inc., 2019.

Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. Improving adversarial robustness via promoting ensemble diversity. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4970–4979, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Sinha, S., Bharadhwaj, H., Goyal, A., Larochelle, H., Garg, A., and Shkurti, F. Dibs: Diversity inducing information bottleneck in model ensembles, 2020.

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

Wen, Y., Tran, D., and Ba, J. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

Xie, Q., Hovy, E. H., Luong, M., and Le, Q. V. Self-training with noisy student improves imagenet classification. *CoRR*, abs/1911.04252, 2019.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2017.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. CutMix: Regularization strategy to train strong classifiers with localizable features. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6022–6031, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

A. Results without BatchEnsemble

We compare (Table 3) performance with AUGMIX for single models (No BatchEnsemble). No clear trends emerge except that always augmenting ($p = 1.0$) leads to better performance on CIFAR10-C and CIFAR100-C. Perhaps surprisingly, augmenting all input images does not hurt i.i.d. accuracy.

Table 3. Comparing performance when using the AUGMIX (referred to as 'AM' in the table) method, with either a Beta distribution to mix augmented images, or a Bernoulli ('Bern.') distribution, with p being the probability of augmenting an input image. 'No B.E.' refers to using a single model. We highlight in bold the best result for each metric.

	CIFAR 10			CIFAR 100		
METHOD	VAL. ERR.	VAL. ECE	VAL. ECE-RMS	VAL. ERR.	VAL. ECE	VAL. ECE-RMS
AM (NO B.E.) BETA	**4.04**	1.44	3.54	22.4	**8.22**	**10.1**
AM (NO B.E.) BERN. ($p = 0.875$)	4.46	**1.35**	3.54	**21.6**	8.42	10.2
AM (NO B.E.) BERN. ($p = 1.0$)	4.30	1.47	**2.90**	21.7	8.33	10.7
	CIFAR 10-C			CIFAR 100-C		
	ERR.	ECE	ECE-RMS	ERR.	ECE	ECE-RMS
AM (NO B.E.) BETA	13.2	**5.36**	**7.06**	37.8	15.7	17.4
AM (NO B.E.) BERN. ($p = 0.875$)	13.1	5.38	7.18	**36.7**	16.1	18.0
AM (NO B.E.) BERN. ($p = 1.0$)	**13.0**	5.41	7.18	**36.7**	**15.8**	**17.7**