RegMixup: Mixup as a Regularizer Can Surprisingly Improve Accuracy and Out-of-Distribution Robustness

Francesco Pinto*
University of Oxford
francesco.pinto@eng.ox.ac.uk

Harry Yang
Meta AI

Ser-Nam Lim
Meta AI

Philip H.S. Torr
University of Oxford

Puneet K. Dokania
University of Oxford & Five AI Ltd.
puneet.dokania@five.ai

Abstract

We show that the effectiveness of the well celebrated Mixup [Zhang et al., 2018] can be further improved if instead of using it as the sole learning objective, it is utilized as an additional regularizer to the standard cross-entropy loss. This simple change not only provides much improved accuracy but also significantly improves the quality of the predictive uncertainty estimation of Mixup in most cases under various forms of covariate shifts and out-of-distribution detection experiments. In fact, we observe that Mixup yields much degraded performance on detecting out-of-distribution samples possibly, as we show empirically, because of its tendency to learn models that exhibit high-entropy throughout; making it difficult to differentiate in-distribution samples from out-of-distribution ones. To show the efficacy of our approach (RegMixup[2]), we provide thorough analyses and experiments on vision datasets (ImageNet & CIFAR-10/100) and compare it with a suite of recent approaches for reliable uncertainty estimation.

1 Introduction

In real-world machine learning applications one is interested in obtaining models that can reliably process novel inputs. However, though deep learning models have enabled breakthroughs in multiple fields, they are known to be unreliable when exposed to samples obtained from a distribution that is different from the training distribution. Usually, the larger the extent of this difference between train and test distributions, the more unreliable these models are. This observation has led to a growing interest in developing deep learning models that can provide *reliable* predictions even when exposed to unseen situations [Liu et al., 2020a,b, Wen et al., 2021, Lakshminarayanan et al., 2017]. Most of these approaches either use expensive ensembles, or propose non-trivial modifications to the neural network architectures in order to obtain reliable models. In most cases, they trade in-distribution performance (accuracy) to perform reliably on: (1) out-of-distribution (OOD) samples; and (2) covariate shift (CS) [Quionero-Candela et al., 2009] samples.

Towards developing practically useful and reliable models, we investigate the well known Mixup [Zhang et al., 2018] as it is extremely popular in improving both a model's accuracy and its robustness [Wen et al., 2021, Hendrycks et al., 2020b]. It has already been observed that Mixup can

*FP continued spending time on this project during his internship at Five AI from June 2021 to March 2022.
[2]Code available at: https://github.com/FrancescoPinto/RegMixup

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

help in retaining good accuracy when the test inputs are affected by superficial variations that do not affect the target label (i.e., they undergo CS) [Hendrycks et al., 2020b]. However, we find that its performance degrades significantly when exposed to completely unseen samples with potentially different labels than the ones it was shown during training (OOD). This is undesirable as it is not guaranteed that the real-world test samples will always belong to one of the categories seen during training, and in such situations, a model should be able to reliably reject those samples instead of making wrong predictions on them. We observe that the primary reason for such poor OOD performance of Mixup is that, because of the way it is trained, it ends up providing high predictive entropy for almost all the samples it receives. Therefore, it becomes difficult for the model to differentiate in-distribution samples from out-of-distribution ones. We would like to highlight that our observation is in contrast to the prior work [Thulasidasan et al., 2019] which suggests that Mixup provides reliable uncertainty estimates for OOD data as well.

We propose a simple yet effective fix to the aforementioned issue with Mixup: we suggest to train the model on an approximate data-distribution that is an *explicit* mixture of both the empirical and the Mixup vicinal approximations to the data-distribution. We call this approach RegMixup. In a nutshell, it simply combines the Empirical Risk Minimization (ERM) [Vapnik, 1991] and Vicinal Risk Minimization (VRM) [Chapelle et al., 2000] objectives together. Implementation wise, along with the Mixup objective on the interpolated samples, it requires adding an additional cross-entropy loss on the clean samples. We provide proper justifications behind this proposal and show that such simple modification can significantly improve the performance of Mixup on a variety of experiments.

We would like to highlight that one of the core strengths of our approach is its **simplicity**. As opposed to the recently proposed techniques to improve uncertainty estimation like SNGP [Liu et al., 2020a] and DUQ [van Amersfoort et al., 2020], it does not require any modifications to the architecture and is extremely simple to implement. *It also does not trade accuracy in order to improve uncertainty estimates*, and is a single deterministic model, hence, extremely efficient compared to the highly competitive Deep Ensembles (DE) [Lakshminarayanan et al., 2017]. Summary of our contributions:

- We provide a simple modification to Mixup that significantly improves its in-distribution, covariate shift, and out-of-distribution performances.
- Through extensive experiments using ImageNet-1K, CIFAR10/100 and their various CS counterparts along with multiple OOD datasets we show that, overall, RegMixup outperforms recent state-of-the-art single-model approaches. In most cases, it outperforms the extremely competitive and expensive DE as well.

2 RegMixup: Mixup as a regularizer

Preliminary on ERM and VRM The principle of risk minimization [Vapnik, 1991] is to estimate a function $f \in \mathcal{F}$ that, for a given loss function $\ell(.,.)$, minimizes the expected risk over the data-distribution $P(\mathbf{x}, \mathbf{y})$. The risk to be optimized is defined as $R(f) = \int \ell(f(\mathbf{x}), \mathbf{y}) dP(\mathbf{x}, \mathbf{y})$. Since the distribution $P(\mathbf{x}, \mathbf{y})$ is unknown, a crude yet widely used approximation is to first obtain a training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ sampled from the distribution P and then obtain f by minimizing the *empirical risk* defined as $R_e(f) = 1/n \sum_{i=1}^n \ell(f(\mathbf{x}_i), \mathbf{y}_i)$. This is equivalent to approximating the entire data-distribution space by a finite n number of Delta distributions positioned at each $(\mathbf{x}_i, \mathbf{y}_i)$, written as $P_e(\mathbf{x}, \mathbf{y}) = 1/n \sum_{i=1}^n \delta_{\mathbf{x}_i}(\mathbf{x}) \delta_{\mathbf{y}_i}(\mathbf{y})$. This approximation to the risk minimization objective is widely known as the *Empirical Risk Minimization (ERM)* [Vapnik, 1991].

ERM has been successfully used in a plenty of real-world applications and undoubtedly has provided efficient and accurate solutions to many learning problems. However, it is straightforward to notice that the quality of such ERM solutions would rely on how closely P_e mimics the true distribution P, and also on the capacity of the function class $\mathcal{F}$. In situations where the function class is extremely rich with very high capacity (like neural networks), and hence prone to undesirable behaviours such as overfitting and memorization, a good approximation to P is generally needed to enforce suitable inductive biases in the model. To this end, for a fixed training dataset, one could potentially fit a richer distribution, instead of a delta distribution, in the vicinity of each input-output pair to estimate a more informed risk computed in a *region* around each sample. This is precisely the principle behind *Vicinal Risk Minimization (VRM)* [Chapelle et al., 2000]. The approximate distribution in this case

can be written as $P_v(\mathbf{x}, \mathbf{y}) = 1/n \sum_{i=1}^{n} P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y})$, where $P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y})$ denotes the user-defined vicinal distribution around the i-th sample[3]. Therefore, the *vicinal risk* boils down to

$$R_v(f) = \frac{1}{n} \sum_{i=1}^{n} \int \ell(f(\mathbf{x}), \mathbf{y}) dP_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y}). \tag{1}$$

When the above integral is intractable, the Monte Carlo estimate with m samples can be obtained as:

$$\int \ell(f(\mathbf{x}), \mathbf{y}) dP_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y}) \approx \frac{1}{m} \sum_{j=1}^{m} \ell(f(\bar{\mathbf{x}}_j), \bar{\mathbf{y}}_j); \;\; (\bar{\mathbf{x}}_j, \bar{\mathbf{y}}_j) \sim P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y}). \tag{2}$$

Several approaches in the deep learning literature can be seen as a special instance of VRM. For example, training a neural network with multiple augmentations is a special case where the augmented inputs are the samples from the unknown vicinal distribution. A widely used application of VRM is the procedure to obtain a *robust base classifier* to design certifiable classifiers. For example, if $P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y}) = P_{\mathbf{x}_i}(\mathbf{x}) \delta_{\mathbf{y}_i}(\mathbf{y})$ and $P_{\mathbf{x}_i}(\mathbf{x})$ is a Gaussian distribution centered at $\mathbf{x}_i$, Eq. (2) can be computed by taking the average loss over gaussian perturbed inputs $\bar{\mathbf{x}}$ while keeping the target labels same. Minimizing such a risk would lead to a classifier that is robust to additive noise bounded within an ℓ_2 ball. This is exactly the procedure that has been widely adopted in the *randomized smoothing* literature in order to obtain a base neural network for which a *certifiable* smooth classifier can be obtained [Lecuyer et al., 2019, Cohen et al., 2019][4]. Below we discuss another highly effective use case of VRM called *Mixup* which is the main focus of this work.

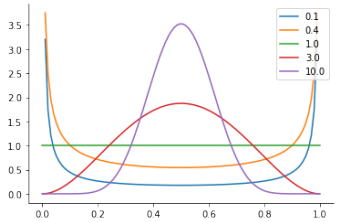

Figure 1: $\texttt{Beta}(\alpha, \alpha)$ pdf for varying α.

Mixup Built on the fundamentals of VRM, the vicinal distribution defined in Mixup [Zhang et al., 2018] is as follows:

$$P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_\lambda[(\delta_{\bar{\mathbf{x}}_i}(\mathbf{x}), \delta_{\bar{\mathbf{y}}_i}(\mathbf{y}))],$$

where $\lambda \sim \texttt{Beta}(\alpha, \alpha) \in [0, 1]$, $\alpha \in (0, \infty)$, $\bar{\mathbf{x}}_i = \lambda \mathbf{x}_i + (1 - \lambda)\mathbf{x}_j$ and $\bar{\mathbf{y}}_i = \lambda \mathbf{y}_i + (1 - \lambda)\mathbf{y}_j$. Note that the vicinal distribution here not only depends on $(\mathbf{x}_i, \mathbf{y}_i)$ but also on another input-output pair $(\mathbf{x}_j, \mathbf{y}_j)$ drawn from the same training dataset. For a fixed α (parameter of the Beta distribution, refer Figure 1), implementing Mixup would require taking multiple Monte Carlo samples[5] for each datapoint (refer Eq. (2)) which can be computationally prohibitive. Therefore, in practice, only one sample ($m = 1$) per Beta distribution per pair of samples from a batch is considered at a time. This procedure, though might seem like a crude approximation to the original objective, has resulted in highly promising results in a variety of applications and is very well accepted in the research community. Without undermining the remarkable success of such a simple approach, below we highlight two of its behavioural characteristics that, as we show, might limit its effectiveness:

- **Small cross-validated $\alpha \ll 1$:** The shape of the vicinal distribution depends on the hyperparameter of the Beta distribution (refer Figure 1), therefore, the values of α decides the strength of the convex interpolation factor λ. Since high values of α would encourage $\lambda \approx 0.5$ resulting in an interpolated $\bar{\mathbf{x}}$ that is perceptually different from $\mathbf{x}$ (*inducing a mismatch between train and test distributions*), the cross-validated value of α for Mixup always turns out to be very small ($\alpha \approx 0.2$) in order to provide good generalization. Such small values of α leads to sharp peaks at 0 and 1 (refer Figure 1). Therefore, effectively, Mixup ends up *slightly* perturbing a clean sample in the direction of another sample even though the vicinal distribution has the potential to explore a much larger interpolation space.
- **High-entropy behaviour:** As mentioned earlier, $m = 1$ in practice, therefore, it is very unlikely that the interpolation factor $\lambda \in \{0, 1\}$ even for small values of α. Thus, the

[3]Note, the original VRM paper uses $P_{\mathbf{y}_i}(\mathbf{y}) = \delta_{\mathbf{y}_i}(\mathbf{y})$ which simply is a special case of this notation.
[4]Based on our understanding, the current literature do not mention this procedure as an instance of VRM.
[5]We do not investigate how m depends on α.

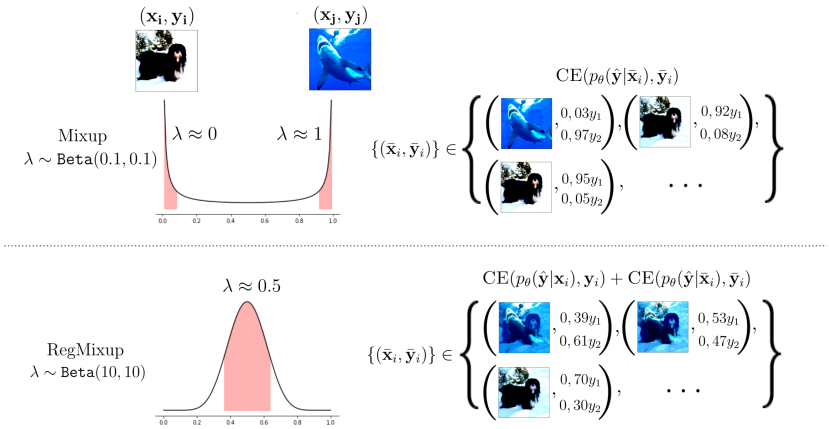

Figure 2: **Mixup vs RegMixup in practice**. Illustration of how the cross-validated α affects the shape of the `Beta` distribution in both cases. Red regions represent 80% of the probability mass. Mixup typically samples $\lambda \approx 0$ or 1, while for RegMixup $\lambda \approx 0.5$. In the first case, one of the two interpolating images dominates the interpolated one; in the latter, a wide variety of images containing features from both the images in the pair are obtained.

model never gets exposed to uninterpolated clean samples during training and hence, it always learns to predict interpolated (or smoothed) labels $\bar{\mathbf{y}}$ for every input. Just like DNNs with cross-entropy loss are overconfident because of their high capacity and target Delta distribution [Guo et al., 2017], DNNs with Mixup turns out to be *relatively less confident* because the network retains its high capacity but observes only target smoothed labels. This underconfident behaviour results in high-entropy for both in-distribution and out-of-distribution samples. This is undesirable as it will not allow predictive uncertainty to reliably differentiate in-distribution samples from out-of-distribution ones.

We validate the consequences of the above limitations of Mixup with the following simple experiment. In Figure 3, we provide heat-maps to show how the entropy of the predictive distribution varies when interpolating samples belonging to different classes. The heat-map is created as follows. First, we train a WideResNet28-10 (WRN) [Zagoruyko and Komodakis, 2016] using the CIFAR-10 (C10) dataset. Then, we randomly choose 1K pairs of samples $\{\mathbf{x}_i, \mathbf{x}_j\}$ from the dataset such that $y_i \neq y_j$[6]. For each pair, via convex combination, we synthesize 20 samples $\bar{\mathbf{x}}$s using equally spaced λs between 0 to 1. The heat-map is then created using all the 20K samples. The intensity of each (λ, H) bin in the heat-map indicates the number of samples in that bin. Note, DNN (trained with vanilla cross-entropy loss) shows low entropy (thus yielding overconfidence) irrespective of where the interpolated sample lies. However, Mixup shows higher entropy for almost all λs (thus yielding lower confidence on in-distribution samples as well). The impact of this behaviour is shown in Table 1. Though Mixup provides improved accuracy compared to DNN for in-distribution (IND) and covariate shift experiments, the high entropy behaviour makes it much worse than DNN when exposed to out-of-distribution detection task. For example, when SVHN is used as the OOD dataset, the performance of Mixup drops by nearly 8.47% compared to DNN. This clearly shows that the distributions of the per-sample entropies on the in-distribution and out-of-distribution sets are harder to separate. However, there is a clear improvement of nearly 5% for covariate shift experiments, implying that Mixup augmentations do improve robustness in this aspect. Note that, in the context

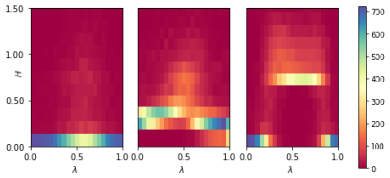

Figure 3: **Heatmaps of the entropy profiles** as the interpolation factor $\lambda \in [0, 1]$ between samples of two classes varies. Left (**DNN**), Middle (**Mixup**), Right (**RegMixup**). Note, RegMixup induces high-entropy barrier separating in-distribution & out-distribution samples.

[6]Note, it is highly *likely* that $\mathbf{y}_i \neq \mathbf{y}_j$ even if we do not impose this constraint as the problems under consideration have many classes.

of calibration, Mixup's confidence reducing behaviour (equivalent to providing higher predictive entropy) was also observed by [Wen et al., 2021].

RegMixup We now provide a very simple modification to Mixup that not only alleviates its aforementioned limitation on detecting OOD samples, but also significantly improves its performance on in-distribution and covariate-shift samples. We propose to use the following approximation to the data-distribution

$$P(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} (\gamma \delta_{\mathbf{x}_i}(\mathbf{x}) \delta_{\mathbf{y}_i}(\mathbf{y}) + (1-\gamma) P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y})),$$

where $P_{\mathbf{x}_i, \mathbf{y}_i}(\mathbf{x}, \mathbf{y})$ is the mixup vicinal distribution and $\gamma \in [0, 1]$ is the mixture weight. *The above approximation is simply an explicit assemble of ERM and VRM* based approximations to data-distribution. Though, in theory, VRM subsumes ERM, we argue that because of the crude approximations needed to prevent increasing the training cost, it does not fully realize its potential. Therefore, explicitly combining them might result in a *practically* more expressive model. We provide extensive experimental evidence to support this hypothesis. Implementation wise, for each sample $\mathbf{x}_i$ in a batch, another sample $\mathbf{x}_j$ is drawn randomly from the same batch to obtain interpolated $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{y}}_i$, and then the following loss is minimized

$$\mathrm{CE}(p_\theta(\hat{\mathbf{y}}|\mathbf{x}_i), \mathbf{y}_i) + \eta \, \mathrm{CE}(p_\theta(\hat{\mathbf{y}}|\bar{\mathbf{x}}_i), \bar{\mathbf{y}}_i), \tag{3}$$

where CE(.,.) denotes the standard cross-entropy loss, the hyperparameter $\eta \in \mathbb{R}_{\geq 0}$, and $p(.)$ the softmax output of a neural network parameterized by θ. Note, dividing Eq. (3) by $(1+\eta)$ is exactly same as using a data-distribution with $\gamma = 1/1+\eta$. In practice, we observe the model's performance to not vary much with η and using $\eta = 1$ (equivalent to $\gamma = 0.5$) to provide highly effective results. Refer to Algorithm 1 for an overview of the RegMixup training procedure.

What does this simple modification bring to the model? (1) VRM now subsumes ERM because the vicinal distribution can perfectly represent the ERM training distribution. This is because of the fact that irrespective of the values of m and α, the model will always be exposed to the clean training samples as well. (2) The interpolation factor λ can potentially explore a much wider space as the presence of clean samples might help in controlling the performance drop due to the train/test distribution shift. Therefore, if $\alpha \ll 1$ was actually the most effective solution, the cross validation would automatically find it.

Practical implications of such simple modification on the behaviour and the performance of the model

- **Large cross-validated** $\alpha \gg 1$: The model is now able to explore strong interpolations because of the additional cross-entropy term over the unperturbed training data. Interestingly, the cross-validated α that we obtained in fact is very high ($\alpha \in \{10, 20\}$, see Appendix A.3 for cross-validation details) leading to $\lambda \approx 0.5$. Therefore, as opposed to the standard Mixup, RegMixup prefers having *strong diverse interpolations* during training. Refer Figure 2 for visualizations.
- **Maximizing a soft proxy to entropy (knowingly unknowing the unknown):** It is straightforward to notice that a value of $\lambda \approx 0.5$ would lead to $\bar{\mathbf{x}}$ that is a heavy mix of two samples (mimicking OOD samples, refer Fig. 2), and the corresponding target label vector $\bar{\mathbf{y}}$ would have almost equal masses corresponding to the labels of the interpolating samples. Therefore, minimizing $\mathrm{CE}(p_\theta(\mathbf{y}|\bar{\mathbf{x}}_i), \bar{\mathbf{y}}_i)$ in this case is equivalent to *maximizing a soft proxy to entropy* defined over the label support of $\mathbf{y}_i$ and $\mathbf{y}_j$ (note, exact entropy maximization would encourage equal probability masses of 0.5 for these labels). We find this observation intriguing as RegMixup naturally obtains a cross-validated α that leads to a maximum likelihood solution having high-entropy on heavily interpolated samples. This is an extremely desirable property as it allows models to reliably differentiate between in and out distribution samples. Thus, Mixup automatically acts as a **regularizer** in this case.

The entropy heat-map in Figure 3 clearly shows that as opposed to DNN and Mixup, the entropy for RegMixup is very low for λ close to either 0 or 1, however, it increases and remains high for all other intermediate interpolation factors, *practically creating an entropy barrier*.

		Cov. Shift	OOD Detection		
Models	**C10** (Test) **Accuracy** (↑)	**C10-C Accuracy** (↑)	**C100 AUROC** (↑)	**SVHN AUROC** (↑)	**T-ImageNet AUROC** (↑)
DNN	96.14	76.60	88.61	96.00	86.44
Mixup	97.01	81.68	83.17	87.53	84.02
RegMixup (Our)	**97.46**	**83.13**	**89.63**	**96.72**	**90.19**

Table 1: In-distribution, covariate shift, and out-of-distribution detection results for WRN trained on CIFAR-10 (C10). C10-C is the corrupted version of C10.

Also, as shown in Table 1, RegMixup provides improvements in both accuracy (in-distribution and covariate-shift) and out-of-distribution robustness. An interesting observation is that Mixup's performance on OOD (T-ImageNet) dropped by 2.4% compared to DNN whereas RegMixup performed 3.75% better than DNN. Thus, in this case, it effectively improved the performance of Mixup by nearly 6.15%.

3 Related Works

Covariate-Shift Robustness It is well known that neural networks performance can severely degrade when exposed to covariate shift [Taori et al., 2016, Recht et al., 2019, Wang et al., 2019]. In image classification, for instance, covariate shift could occur due to changes in environmental conditions [Hendrycks and Dietterich, 2019] and the image capturing process, data sampling [Recht et al., 2019, 2018, Lu et al.], stylised representations of the objects (e.g. in paintings or drawings) [Wang et al., 2019, Hendrycks et al., 2021] etc. In the field of out-of-distribution generalization, many techniques assume the availability of sets of samples presenting such variations at training time [Shi et al., 2021, Yao et al., 2022, Wang et al., 2021], other techniques suggest leveraging augmentations to mimic a set of variations that would allow the model to learn features that generalize better [Hendrycks et al., 2020b,a] or self-supervision [Hendrycks et al., 2019a, Shen et al., 2022].

Out-Of-Distribution Detection While some authors suggest introducing an OOD reject option [Geifman and El-Yaniv, 2019], the most recent literature tends to focus on leveraging a threshold classifier on an uncertainty measure computed on top of the predictive distribution of a network. For the latter approach, it is therefore convenient for a model to produce high uncertainty predictions when facing OOD samples, and low uncertainty predictions when fed in-distribution samples [Liu et al., 2020a]. Within this family of approaches, the most popular ones are the expensive Deep Ensembles (DE) [Lakshminarayanan et al., 2017]. Note, the computational cost of DE scales linearly with the number of members, and also recent works have observed a few limitations of naively using ensembles directly [Wen et al., 2021, Ashukha et al., 2020]. There are several efficient variants of DE where the ensemble members are obtained either by processing multiple inputs and multiple outputs at the same time [Havasi et al., 2021], adding member-specific components to the network [Wen et al., 2020], or by finding optima in distinct basins of low loss [Huang et al., 2017a]. Significant attention has also been paid to Bayesian techniques where either stochastic gradient descent is being modified to perform approximate inference [Chen et al., 2014, Zhang et al., 2020, Durmus et al., 2016] or an approximation to Gaussian Processes is being utilized (specifically, we focus on SNGP [Liu et al., 2020a]), or Bayesian Logistic Regression is being used during inference via Laplace approximation (specifically, we focus on KFAC-LLLA [Kristiadi et al., 2020]). Techniques utilizing distance functions in the feature space have also been suggested [van Amersfoort et al., 2020].

Model calibration Modern NNs have been shown to be miscalibrated [Guo et al., 2017], i.e. they exhibit a mismatch between the model's confidence and its accuracy. It has been observed that temperature scaling [Guo et al., 2017] or replacing the typical cross-entropy loss [Chung et al., 2021, Thulasidasan et al., 2019, Mukhoti et al., 2020] can be highly effective to reduce this mismatch. Also, ensemble learning has been observed to help in reducing miscalibration [Lakshminarayanan et al., 2017, Wen et al., 2021, Rahaman and Thiery, 2020]. However, it has been shown that good calibration on in-distribution data does not necessarily imply good calibration performance under covariate-shift [Ovadia et al., 2019].

4 Experiments

Datasets and Network Architectures We employ the widely used WideResNet28-10 (WRN) [Zagoruyko and Komodakis, 2016] and ResNet50 (RN50) [He et al., 2016] architectures. We train them on CIFAR-10 (C10) and CIFAR-100 (C100) datasets. We employ RN50 to perform experiments on ImageNet-1K [Deng et al., 2009] dataset. We report the average of all the metrics computed on 5 seeds. For further details about the code base and the hyperparameters, refer to Appendix A.

For **Covariate Shift (CS)** experiments on models trained on C10 and C100, we resort to the widely used CIFAR10-C (C10-C) and CIFAR100-C (C100-C) datasets, corrupted versions of C10 and C100 [Hendrycks and Dietterich, 2019]. These datasets are made by applying 15 synthetically

generated but *realistic corruptions* at 5 degrees of intensity on the test sets of C10 and C100, respectively. For CIFAR-10, we also use the CIFAR-10.1 (C10.1) [Recht et al., 2018] and CIFAR-10.2 (C10.2) [Lu et al.] datasets designed to test the generalization of CIFAR-10 classifiers to *natural covariate shifts*. To the best of our knowledge there is no such analogous dataset for CIFAR-100. For ImageNet-1K experiments, we use the widely considered ImageNet-A (A) [Hendrycks et al., 2019b], ImageNet-R (R) [Hendrycks et al., 2021], ImageNetv2 (V2) [Recht et al., 2019], and ImageNet-Sketch (SK) [Wang et al., 2019] datasets for covariate shift experiments.

For **Out-of-Distribution (OOD)** detection, following SNGP [Liu et al., 2020a], we use C100 and SVHN Netzer et al. [2011] as OOD datasets for models trained on C10. Similarly, for models trained on C100, we use C10 and SVHN as OOD datasets. Additionally, we also consider the Tiny-ImageNet (T-ImageNet) dataset [Le and Yang, 2015] as OOD set in both the cases. For models trained on ImageNet-1K, we use ImageNet-O (O) [Hendrycks et al., 2019b] as the OOD dataset.

Methods considered for comparisons Besides the natural comparison of our method with Mixup [Zhang et al., 2018] and networks trained via ERM on the standard cross-entropy loss (which we will refer to as DNN), we consider several other methods from the OOD detection and CS literature. For models trained on C10 and C100, we consider:

- DNN-SN and DNN-SRN: taking inspiration from [Liu et al., 2020a], we consider DNN models trained with Spectral Normalization (SN) [Miyato et al., 2018a] and Stable Rank Normalization (SRN) [Sanyal et al., 2020] to control the bi-Lipschitz constant of the networks, which has been shown to affect the generalization properties of neural networks.
- SNGP: Spectrally Normalized Gaussian Process [Liu et al., 2020a].
- DUQ: Deterministic Uncertainty Quantification [van Amersfoort et al., 2020].
- KFAC-LLLA: KFAC-Laplace Last Layer Approximation [Kristiadi et al., 2020] makes a model Bayesian at test time by taking the Laplace approximation of the last layer [Ritter et al., 2018]. We provide a simple outline of this approach in Appendix D.
- AugMix [Hendrycks et al., 2020b]: A data augmentation technique that applies randomized augmentations to an input while enforcing them to be consistent during training. With the recommended hyperparameters in the paper, it is almost $4\times$ slower than DNN during training while having the same inference requirements.
- DE: Deep Ensembles [Lakshminarayanan et al., 2017] with 5 members, requiring $5\times$ more compute than most single-model approaches such as DNN.

We would like to mention that, compared to vanilla DNN, our approach (RegMixup) is almost $1.5\times$ slower and Mixup is about $1.2\times$ slower during training, while having the same inference requirements. Due to the high compute requirements, for ImageNet-1K we consider DNN, Mixup and the two other strongest baselines: AugMix and DE. We also cross-validate the hyperparameters on a 10% split of the test set, which is removed at test time.

Few missing experiments: Below we provide extensive experiments for proper benchmarking. The number of datasets and architectures we use lead to many combinations, a few of which we were not able to produce good results for (even after extensive hyperparameter search), hence, some of the entries in the tables are missing. For example, we could not make DUQ work on C100 as it exhibited unstable behaviour. We could not produce promising results for SNGP on RN50 CIFAR experiments using their official implementation. Similarly for AugMix RN50 experiments on C10 and C100. We chose not to report these suboptimal numbers. Further details can be found in Appendix A.

Table entries: Bold represents the best among all the single-model approaches, and underlined represents the best among all including the expensive Deep Ensembles.

Note, we do not consider methodologies requiring access to an external dataset during training (either for CS or OOD) as not only this would be an unfair comparison[7], but we believe assuming such knowledge is against the goal of this work, which is to develop models robust to unknown scenarios.

[7] since all the methods we consider do not leverage this information. RegMixup only relies on in-distribution training data, just like other approaches, and makes no assumption on the type of CS nor on the OOD inputs.

4.1 RegMixup Improves Accuracy on In-distribution and Covariate-shift Samples

Small-scale (CIFAR) Experiments on In-distribution Data In Table 2, we compare the accuracy of various approaches on the in-distribution test sets of C10 and C100, respectively. Clearly,

- RegMixup outperforms Mixup in all these experiments. In fact, RegMixup is the best performing one among all the single-model approaches.
- These improvements are non-trivial. For instance, on WRN trained on C100, it outperforms DNN and Mixup by 1.67% and 0.65%, respectively. It outperforms SNGP with a significant margin of 4.05%.

Note how the single-model approaches specifically designed to provide reliable predictive uncertainty estimations (for example, SNGP, DUQ, KFAC-LLLA) underperform even compared to the vanilla DNN in terms of in-distribution accuracy. In order to provide improved uncertainty estimates (as we will soon show), they trade clean data accuracy. This type of behaviour is not observed in RegMixup.

	IND Accuracy			
	WRN28-10		RN50	
Methods	C10 (Test) Accuracy (↑)	C100 (Test) Accuracy (↑)	C10 (Test) Accuracy (↑)	C100 (Test) Accuracy (↑)
DNN	96.14	81.58	95.19	79.19
Mixup	97.01	82.60	96.05	80.12
RegMixup (**our**)	**97.46**	**83.25**	**96.71**	**81.52**
DNN-SN	96.22	81.60	95.20	79.27
DNN-SRN	96.22	81.38	95.39	78.96
SNGP	95.98	79.20	-	-
DUQ	94.7	-	-	-
KFAC-LLLA	96.11	81.56	95.21	79.41
Augmix	96.40	81.10	-	-
DE (5×)	96.75	83.85	96.23	82.09

Table 2: Accuracies (%) on IND samples for models trained on C10 and C100

Small-scale (CIFAR) Covariate Shift Experiments For C10-C and C100-C, as typical in the literature, we report the accuracy averaged over all the corruptions and degrees of intensities in Table 3. It is evident that our approach produces a remarkable improvement in the average accuracy compared to **all** the baselines (except AugMix, we discuss later why that might be the case). For instance, for C100-C WRN experiments, our method achieves an accuracy improvement of **6.9%** over DNN, of **3.86%** over DE, and of **2.45%** over Mixup. Similarly, for C10-C WRN, our method achieves an improvement of almost **6.53%** over DNN, of **4.81%** over DE, and of **1.45%** over Mixup.

For natural covariate shift datasets C10.1 and C10.2 as well, RegMixup outperforms **all** the baselines (including AugMix). For instance, on C10.2, it obtains an improvement of **3.26%** over DNN, of **1.5%** over Mixup, and of **2.46%** over the expensive DE.

Why AugMix performs extraordinarily well on synthetically corrupted C10-C and C100-C but not on natural distribution shift C10.1 and C10.2? Looking at Table 3 one can observe AugMix's extremely good performance on the C10-C and C100-C. However, at the same time, the model is underperforming with respect to RegMixup on C10.1 and C10.2. Similarly, AugMix is outperformed by RegMixup on all ImageNet CS experiments (and OOD as will be shown soon). This seems to suggest that although the augmentations used during the training of AugMix are not exactly same as that of the corrupted test dataset, they tend to benefit from synthetic forms of covariate shifts, hence the dramatic improvement in these particular scenarios.

	Covariate Shift Accuracy							
	WRN28-10				R50			
Methods	C10-C Accuracy (↑)	C10.1 Accuracy (↑)	C10.2 Accuracy (↑)	C100-C Accuracy (↑)	C10-C Accuracy (↑)	C10.1 Accuracy (↑)	C10.2 Accuracy (↑)	C100-C Accuracy (↑)
DNN	76.60	90.73	84.79	52.54	75.18	88.58	83.31	50.62
Mixup	81.68	91.29	86.55	56.99	78.63	90.03	84.61	53.96
RegMixup (**our**)	83.13	**92.79**	**88.05**	59.44	**81.18**	**91.58**	**86.72**	**57.64**
DNN-SN	76.56	91.01	84.72	52.61	74.88	88.26	82.96	50.55
DNN-SRN	77.21	90.88	85.24	52.54	75.40	88.61	83.49	50.48
SNGP	78.37	90.80	84.95	57.23	-	-	-	-
DUQ	71.6	-	-	50.4	-	-	-	-
KFAC-LLLA	76.56	90.68	84.68	52.57	75.18	88.34	83.50	50.85
AugMix	**90.02**	91.6	85.9	**68.15**	-	-	-	-
DE (×5)	78.32	92.17	85.59	55.58	77.63	90.05	85.00	53.91

Table 3: Accuracies (%) on covariate shifted samples for models trained on C10 and C100.

	IND Acc.	Covariate Shift Acc				OOD Det.
	ImageNet-1K Acc (↑)	R Acc (↑)	A Acc (↑)	V2 Acc (↑)	Sk Acc (↑)	O AUROC (↑)
DNN	76.60	36.41	2.76	64.53	24.72	55.97
Mixup	77.15	39.05	3.29	64.58	26.34	55.54
RegMixup (**our**)	**77.68**	**39.76**	**5.96**	**65.66**	**26.98**	**57.05**
AugMix	76.88	38.29	2.63	64.94	25.61	56.91
DE (5×)	78.22	38.94	2.11	66.68	27.03	53.29

Table 4: ImageNet accuracies (%) on IND and CS samples, and OOD detection performance.

Large-scale (ImageNet-1K) Experiments on In-distribution Data As shown in Table 4, RegMixup scales to ImageNet-1K and exhibits improved accuracy with respect to both Mixup, DNN, and AugMix. In particular, it is **1.08%** better than DNN, **0.53%** better than Mixup and **0.80%** better than AugMix. DE in this case is the best performing one.

Large-scale (ImageNet-1K) Covariate Shift Experiments In Table 4 we report the results for common ImageNet-1K covariate-shift test sets. As it can be seen, RegMixup is either the best

	WRN28-10						RN50					
	CIFAR10 (In-Distribution)			CIFAR100 (In-Distribution)			CIFAR10 (In-Distribution)			CIFAR100 (In-Distribution)		
Out-of-Distribution	**C100**	**SVHN**	**T-ImageNet**	**C10**	**SVHN**	**T-ImageNet**	**C100**	**SVHN**	**T-ImageNet**	**C10**	**SVHN**	**T-ImageNet**
Methods	**AUROC** (↑)			**AUROC** (↑)			**AUROC** (↑)			**AUROC** (↑)		
DNN	88.61	96.00	86.44	81.06	79.68	80.99	88.61	93.20	87.82	79.33	82.45	79.89
Mixup	83.17	87.53	84.02	78.37	78.68	80.61	84.24	89.40	84.89	77.02	76.86	80.14
RegMixup (**our**)	89.63	**96.72**	**90.19**	**81.27**	**89.32**	**83.13**	**89.63**	**95.39**	**90.04**	**79.44**	**88.66**	**82.56**
DNN-SN	88.56	95.59	87.71	81.10	83.43	82.26	88.19	93.46	87.55	79.20	80.78	79.90
DNN-SRN	88.46	96.12	87.43	81.26	85.51	82.41	88.82	93.54	87.82	78.77	82.39	79.70
SNGP	**90.61**	95.25	90.01	79.05	86.78	82.60	-	-	-	-	-	-
KFAC-LLLA	89.33	94.17	87.81	81.04	80.32	81.57	89.54	93.13	88.32	79.30	82.80	80.17
Aug-Mix	89.78	91.3	88.99	81.10	76.64	80.56	-	-	-	-	-	-
DE (5×)	91.25	97.53	89.52	83.26	85.07	83.40	91.38	96.90	90.5	81.93	85.08	82.15

Table 5: Out-of-distribution detection results (%) for WideResNet28-10 and ResNet50 for models trained on C10 and C100. See Appendix A for the cross-validated hyperparameters.

performing one among all the single-model approaches, or it is the absolute winner including DE. For instance, on ImageNet-A, RegMixup performs **2.67%** better than Mixup and **3.20%** better than DNN. Similarly, on ImageNet-V2 it performs **1.08%** better than Mixup and **1.13%** more than DNN. RegMixup also outperforms AugMix on all the considered datasets, while is outperformed by DE on ImageNet-V2 (by **1.02%**) and performs competitively on ImageNet-SK.

These experiments clearly show the strong generalization of RegMixup under various in-distribution and CS experiments. They also show that it does not trade clean data accuracy to do so.

4.2 Out-of-Distribution Detection Experiments

Following the standard procedure [Liu et al., 2020a], we report the performance in terms of AUROC[8] for the binary classification problem between in- and out-distribution samples. The predictive uncertainty of the model is typically used to obtain these curves. Given an uncertainty measure (normally entropy, refer Appendix B for an extensive discussion), it is important for models to be more uncertain on OOD samples than on in-distribution samples to be able to distinguish them accurately. This behaviour would lead to a better AUROC.

We report the OOD detection results for small-scale experiments (CIFAR 10/100) in Table 5, and for large-scale experiments (ImageNet) in Table 4.

Compared to the single-model approaches, it is clear that RegMixup outperforms all the baselines in both small-scale and large-scale experiments, except only in one situation (C100 as OOD, trained on WRN) where SNGP outperformed it by an AUROC of 0.98.

In fact, on ImageNet experiments, RegMixup outperformed all the baselines including DE.

We would also like to highlight that RegMixup provides significantly better performance compared to Mixup on all these experiments. For example, the improvement is more than 9% when SVHN is treated as the OOD dataset for WRN trained on either C10 or C100 (refer Table 5). Similarly, we can observe dramatic improvements on other experiments as well.

4.3 Calibration on In-distribution and Covariate-shift datasets

Additionally, we provide the calibration performance of various competitive approaches. Briefly, calibration quantifies how similar a model's confidence and its accuracy are [Osborne, 1991]). To measure it, we employ the recently proposed Adaptive ECE (AdaECE) [Mukhoti et al., 2020]. For all the methods, the AdaECE is computed after performing temperature scaling [Guo et al., 2017] with a cross-validated temperature parameter. We also provide the AdaECE without temperature scaling in Appendix C. For completeness, we also report the ECE in Appendix F.

In terms of calibration on in-domain test sets (refer Tables 6 and 7), our method either remarkably improves the AdaECE with respect to Mixup and DNN, or performs competitively (on ImageNet-1K).

Under covariate shift (refer Tables 7 and 8), on corrupted inputs, RegMixup underperforms with respect to Mixup on C10-C, but not on C100-C. On all other C10 covariate shift datasets, RegMixup

[8] Area Under Receiver Operating Characteristic curve.

outperforms both Mixup and DNN. Considering also the other baselines, except for the case of C10-C (in which AugMix significantly outperforms any other baseline on WRN28-10), our method provides the best calibration in all other cases. For example, on C100-C experiments on WRN28-10, in terms of AdaECE, RegMixup obtains a **4.43%** improvement over DE, **2.52%** over Mixup, and **2.47%** over SNGP. Though RegMixup outperformed all other approaches in 12 scenarios out of total 17 presented here, it is clear that there is no single method that outperforms any other in all the considered settings.

5 Conclusive Remarks

We proposed RegMixup, an extremely simple approach that combines Mixup with the standard cross-entropy loss. We conducted a wide range of experiments and showed that RegMixup significantly improved the reliability of uncertainty estimates of deep neural networks, while also provided a notable boost in the accuracy. We showed that RegMixup did not just outperform Mixup, it also outperformed most recent state-of-the-art approaches in providing reliable uncertainty estimates.

We hope that our work opens possibilities to explore situations where ERM and VRM can explicitly be combined together for practical benefits. An example would be label smoothing [Müller et al., 2019] that can be seen as an instance of VRM where the vicinal distribution is over the labels and the marginal distribution of the input samples (e.g., images) is approximated using deltas. In Appendix G we conduct a preliminary analysis using CutMix Yun et al. [2019] and recent techniques to train Vision Transformers Dosovitskiy et al. [2021] that alternate between Mixup and CutMix. Another possible future direction would be to use bi-modal vicinal distribution along with importance sampling in order to ensure that the unperturbed samples as well are used during training with high probability. The observation that RegMixup ends up acting as a soft proxy to entropy maximizer for interpolated samples, a potential extension of our work regards the possibility of exploring different interpolation techniques to enforce high entropy behaviour on those regions of the input space.

	IND			
	WRN28-10		**RN50**	
Methods	**C10 (Test)**	**C100 (Test)**	**C10 (Test)**	**C100 (Test)**
	AdaECE (↓)		**AdaECE (↓)**	
DNN	1.34	3.84	1.45	2.94
Mixup	1.16	1.98	2.17	7.47
RegMixup (**our**)	**0.50**	**1.76**	**0.94**	**1.53**
SNGP	0.87	1.94	-	-
Augmix	1.67	5.54	-	-
DE (5×)	1.04	3.29	1.28	2.98

Table 6: CIFAR IND calibration performance (%).

	IND	**Covariate Shift**			
	ImageNet-1K (Test)	**ImageNet-R**	**ImageNet-A**	**ImageNet-V2**	**ImageNet-Sk**
	AdaECE (↓)	**AdaECE (↓)**	**AdaECE (↓)**	**AdaECE (↓)**	**AdaECE (↓)**
DNN	1.81	13.56	44.90	4.13	14.48
Mixup	**1.29**	12.08	44.63	4.28	15.26
RegMixup (**our**)	1.37	13.30	**41.18**	**3.38**	15.35
AugMix	2.05	**11.24**	42.83	3.94	**14.26**
DE (5×)	1.38	13.55	42.88	4.02	17.32

Table 7: ImageNet calibration performance on IND and CS datasets (%).

	Covariate Shift							
	WRN28-10				**R50**			
Methods	**C10-C**	**C10.1**	**C10.2**	**C100-C**	**C10-C**	**C10.1**	**C10.2**	**C100-C**
	AdaECE (↓)				**AdaECE (↓)**			
DNN	12.62	4.13	8.81	9.94	12.29	4.36	8.89	19.76
Mixup	7.93	4.39	7.44	10.45	**10.75**	5.72	10.59	12.63
RegMixup (**our**)	9.08	**2.57**	**6.83**	**7.93**	11.37	**2.89**	**6.74**	**11.47**
SNGP	11.34	4.36	8.33	10.43	-	-	-	-
AugMix	**4.56**	3.23	8.33	12.15	-	-	-	-
DE (5 ×)	10.31	2.60	7.50	12.36	12.68	4.10	6.94	12.36

Table 8: CIFAR CS calibration performance (%).

Acknowledgements

This work is supported by the UKRI grant: Turing AI Fellowship EP/W002981/1 and EPSRC/MURI grant: EP/N019474/1. We would like to thank the Royal Academy of Engineering, FiveAI and Meta AI. Francesco Pinto's PhD is funded by the European Space Agency (ESA). Meta AI authors are neither supported by the UKRI grants nor have any relationships whatsoever to the grant. Thanks to John Redford, Jon Sadeghi, Kemal Oksuz, Zygmunt Lenyk, Guillermo Ortiz-Jimenez, Amartya Sanyal, Pau de Jorge, and David Lopez-Paz for their valuable comments on the paper.

References

Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020.

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

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *13th International Conference on Neural Information Processing Systems*, 2000.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, 2014.

Youngseog Chung, Willie Neiswanger, Ian Char, and Jeff Schneider. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. In *Advances in Neural Information Processing Systems*, 2021.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

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`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Alain Durmus, Umut Simsekli, Eric Moulines, Roland Badeau, and Gaël RICHARD. Stochastic gradient richardson-romberg markov chain monte carlo. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org, 2016.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning*, pages 2151–2159. PMLR, 2019.

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.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *CVPR*, 2017.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=OGg9XnKxFAH`.

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.

D Hendrycks, S Basart, N Mu, S Kadavath, and others. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv*, 2020a.

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

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2019a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. July 2019b.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020b.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

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`.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free, 2017a.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017b.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, 2015.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in ReLU networks. In *Proceedings of the 37th ICML*, pages 5436–5446, 2020.

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*, 2017.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

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 2020a.

Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection, 2020b.

Shangyun Lu, 1 Bradley Nott, and 1 Aaron Olson. Harder or different? a closer look at distribution shift in dataset reproduction. `http://www.gatsby.ucl.ac.uk/~balaji/udl2020/accepted-papers/UDL2020-paper-101.pdf`. Accessed: 2021-11-10.

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 2018a.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=B1QRgziT-.

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.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In H Wallach, H Larochelle, A Beygelzimer, F dAlché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 4694–4703. Curran Associates, Inc., 2019.

Yuval Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Christine Osborne. Statistical calibration: A review. *International Statistical Review / Revue Internationale de Statistique*, 59(3):309–336, 1991. ISSN 03067734, 17515823. URL http://www.jstor.org/stable/1403690.

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.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN 0262170051.

Rahul Rahaman and Alexandre H. Thiery. Uncertainty quantification and deep ensembles, 2020.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to CIFAR-10? June 2018.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *CoRR*, abs/1902.10811, 2019. URL http://arxiv.org/abs/1902.10811.

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 *Advances in Neural Information Processing Systems*, 2018.

Kendrick Shen, Robbie Jones, Ananya Kumar, Sang Michael Xie, Jeff Z. HaoChen, Tengyu Ma, and Percy Liang. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation, 2022. URL https://arxiv.org/abs/2204.00570.

Yuge Shi, Jeffrey Seely, Philip H. S. Torr, N. Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

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 *Advances in Neural Information Processing Systems*, 2019.

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

V. Vapnik. Principles of risk minimization for learning theory. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, page 831–838, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558602224.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*, 2021.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Sklf1yrYDr`.

Yeming Wen, Ghassen Jerfel, Rafael Muller, Michael W Dusenberry, Jasper Snoek, Balaji Lakshminarayanan, and Dustin Tran. Combining ensembles and data augmentation can harm your calibration. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=g11CZSghXyY`.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation, 2022. URL `https://openreview.net/forum?id=zXne1klXIQ`.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *ICCV*, 2019.

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`.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkeS1RVtPS`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the "Methods considered for Comparison" section
   (c) Did you discuss any potential negative societal impacts of your work? [No] Not expected
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is on github. The appendix contains an explicative pseudocode

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] They are specified either in the Experiments section of the main paper or the Appendix

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [No]

   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Algorithm 1 RegMixup training procedure

input Batch $\mathcal{B}$, α, θ_t
  $\bar{\mathcal{B}} \leftarrow \emptyset$
  $\lambda_0 \sim \texttt{Beta}(\alpha, \alpha)$
  for $\forall(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{B}$ **do**
    randomly select $(\mathbf{x}_j, \mathbf{y}_j)$ from $\mathcal{B} \backslash (\mathbf{x}_i, \mathbf{y}_i)$
    $\bar{\mathcal{B}} \leftarrow \bar{\mathcal{B}} \cup (\lambda_0 \mathbf{x}_i + (1-\lambda_0)\mathbf{x}_j, \lambda_0 \mathbf{y}_i + (1-\lambda_0)\mathbf{y}_j)$
  end for
  $\mathcal{L} = \mathrm{CE}(\mathcal{B}) + \mathrm{CE}(\bar{\mathcal{B}})$ *// Loosely speaking, compute cross-entropy loss on both the batches*
  return θ_{t+1} obtained by updating θ_t by optimizing the above loss

A Experimental Details

A.1 Code-base

The **RegMixup** training procedure is outlined in Algorithm 1.

For fair comparisons, when training on C10 and C100, we developed our own code base for all the approaches (except SNGP, DUQ and AugMix) and performed an extensive hyperparameter search to obtain the strongest possible baselines.

We would like to highlight that it was not easy to make a few recent state-of-the-art approaches work in situations different from the ones they reported in their papers as these approaches mostly required non-trivial changes to the architectures and additional sensitive hyperparametes. We also observed that their performances did not easily translate to new situations. Below we highlight few of these issues we faced and the measures we took for comparisons.

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 the training procedure seemed to be *unstable*. For this reason, *wherever possible*, we borrowed the 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 SNGP, we used the publicly available code following exactly the same procedure as mentioned in their original paper. The code *diverges slightly* from the procedure described in their paper, hence the slight differences in the performance. The only 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 SNGP RN50 experiments, we tried running the official implementation on C10 and C100, but we could not make SNGP converge to SOTA accuracy values. The authors of SNGP did not provide experiment results on C10 and C100 on RN50. Hence we decided *not to report* these experiments for SNGP.

For the KFAC-LLLA we leverage the official repository[9] [Hobbhahn et al., 2021] and the Backpack library [Dangel et al., 2020] for the computation of the Kronecker-Factored Hessian.

For AugMix, we used their code base and the exact training procedure. AugMix seems to be sensitive to hyperparameters of the training procedure as we could could not get the considered architectures to converge to acceptable accuracy levels under the training regime we used for **all** other baselines. Even with the recipes provided in the AugMix paper, we could not get it to converge to competitive accuracy levels when using RN50 on C10 and C100 hence we decided *not to report* these experiments for AugMix.

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

A.2 Optimization

For **C10 and C100** training, we use SGD with Nesterov momentum 0.9 for 350 epochs and a weight decay of 5×10^{-4}. For WRN, we apply a dropout $p = 0.1$ at train time. For all our experiments we set the batch size to 128[10]. At training time, we apply standard augmentations `random crop` and `horizontal flip` similar to [Liu et al., 2020a]). The data is appropriately normalized before being fed to the network both at train and test time.

For **ImageNet-1K** training, we use SGD with momentum for 100 epochs, learning rate 0.1, cosine learning scheduler, weight decay of 1×10^{-4}, batch size 128 and image size 224×224. We use `color jitter`, `random horizontal flip` and `random crop` for augmentation. We leverage the `timm` library for training [Wightman, 2019] all the considered methods with Automatic Mixed Precision to accelerate the training.

A.3 Hyperparameters

- For DNN-SN and DNN-SRN the spectral norm clamping factor (maximum spectral norm of each linear mapping) $c \in \{0.5, 0.75, 1.0\}$ and 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 $c = 1.0$). Refer to Miyato et al. [2018b] and [Sanyal et al., 2020] for details about these hyperparameters.
- For Mixup, we consider a wide range of Beta distribution hyperparameter $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 1, 5, 10, 20\}$.
- For RegMixup we consider the Beta distribution hyperparameters to be $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 1, 5, 10, 15, 20, 30\}$, and the mixing weight $\eta \in \{0.1, 1, 2\}$.
- For KFAC-LLLA we take 1000 samples from the distribution. 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. We provide an overview of the KFAC-LLLA in Appendix D.
- 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 0.001). 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 cross-validated hyperparameters are reported in Table 9. The cross-validation is performed with stratified-sampling on a 90/10 split of the training set to maximise accuracy[11] on C10 and C100. For ImageNet, we split the test set using the same proportion to obtain the validation set, which is then removed from the test set during evaluation. It is important to observe that:

- Cross-validating hyperparameters based solely on the ECE can prefer models with lower accuracy but better calibration. However, a method improving calibration should avoid degrading accuracy.
- Hyperparameters should not be cross-validated based on CS experiments and OOD detection metrics as they these datasets should be unknown during the training and hyperparameter selection procedure as well.

The results of the cross-validation for RegMixup for CIFAR-10, CIFAR-100 can be found in Table 14. To support our claim that Mixup tends to prefer lower α values, we do not only report the cross-validation accuracy on Mixup for the experiments in the main paper (Table 12) but we also report them for two other popular architectures on CIFAR-10 and CIFAR-100: DenseNet-121 Huang et al. [2017b] and PyramidNet200 Han et al. [2017] (Table 13).

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

[11] 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

Train Data/ Architecture	Hyperparas	C10		C100		ImageNet
		WRN	R50	WRN	R50	
DNN	T	1.32	1.51	1.33	1.42	1.19
DNN-SN	c	0.5	0.5	0.5	0.5	-
	T	1.42	1.51	1.21	1.42	-
DNN-SR	r	0.3	0.3	0.3	0.3	-
	T	1.33	1.41	1.22	1.42	-
DE	T	1.31	1.42	1.11	1.21	1.29
SNGP	T	1.41	-	1.52	-	-
Mixup	α	0.3	0.3	0.3	0.3	0.1
	T	0.73	0.82	1.09	1.21	1.06
RegMixup	$\eta = 1, \alpha$	20	20	10	10	10
	T	1.12	1.31	1.23	1.21	1.14
KFAC-LLLA	#samples	1000	1000	1000	1000	
	σ_0	1	0.6	4	0.1	-

Table 9: Cross-validated hyperparameters. Note, T and σ_0 are cross-validated by minimizing the ECE. All other hyperparameters have been tuned to maximise the accuracy.

B Existing Uncertainty Measures

There are various uncertainty measures and there is no clear understanding on which one would be more reliable. In our experiments we considered the following metrics and chose the one best suited for each method in order to create the strongest possible baselines. Let K denote the number of classes, $\mathbf{p}_i$ the probability of i-th class, and $\mathbf{s}_i$ the `logit` of i-th class. Then, these uncertainty measures can be defined as:

- **Entropy**: $H(\mathbf{p}(\mathbf{x})) = -\sum_{i=1}^{K} \mathbf{p}_i \log \mathbf{p}_i$.
- **Dempster-Shafer** [Sensoy et al., 2018]: $\texttt{DS}(\mathbf{x}) = {}^{K}/_{(K+\sum_{i=1}^{K} \exp(\mathbf{s}_i))}$.
- **Energy**: $E(\mathbf{x}) = -\log \sum_{i=1}^{K} \exp(\mathbf{s}_i)$ (ignoring the temperature parameter). This metric was used in [Liu et al., 2020b] for OOD.
- **Maximum Probability Score**: $\texttt{MPS}(\mathbf{x}) = \max_i \mathbf{p}_i$.
- **Feature Space Density Estimation** (FSDE): Assuming that the features of each class follow a Gaussian distribution, there are several ways one can estimate the *belief* of a test sample belonging to in-distribution data and treat it as a measure of uncertainty. One such approach is to compute the Mahalanobis score $\arg\min_{i \in y} (\phi(\mathbf{x}) - \mu_i)^T \Sigma_i^{-1} (\phi(\mathbf{x}) - \mu_i)$, where μ_i and Σ_i are class-wise mean and the covariance matrices of the *train* data, and $\phi(\mathbf{x})$ is the feature vector.

In the main paper, we report the OOD detection performance using the `DS` *score (as it provided slightly improved performance in most cases), except when it damages the performance of a method (e.g. Mixup) or when it does not yield improvements (e.g. KFAC-LLLA). In these situations we use the entropy as the uncertainty measure.*

Remarks regarding various metrics: We would like to highlight a few important observations that we made regarding these metrics. **(1)** `DS` **and** E **are equivalent** as they are both decreasing functions of $\sum_{i=1}^{K} \exp(\mathbf{s}_i)$, and since $\log$ does not modify the monotonicity, both will provide the same ordering of a set of samples. Hence, will give the same AUROC values. **(2)** We observed `DS` and H to perform similarly to each other except in a few situations where `DS` provided slightly better results. **(3)** `MPS`, in many situations, was slightly worse. **(4)** We found Gaussian assumption based density estimation to be **unreliable**. Though it provided extremely competitive results for C10 experiments, sometimes slightly better than the `DS` based scores, it performed very poorly on C100. We found this score to be highly unstable as it involves large matrix inversions. We applied the well-known tricks such as perturbing the diagonal elements and the low-rank approximation with high variance-ratio, but the results were sensitive to such stabilization and there is no clear way to cross-validate these hyperparameters.

Methods	Clean AdaECE (↓)	CIFAR-10-C AdaECE (↓)	CIFAR 10.1 AdaECE (↓)	CIFAR 10.2 AdaECE (↓)
C10 R50				
DNN	3.02	17.30	7.39	12.24
Mixup	2.87	**11.35**	**4.05**	**7.72**
RegMixup (**Ours**)	1.40	11.51	4.15	8.23
DE (5×)	2.10	13.99	6.23	10.33
C10 WRN				
DNN	2.27	15.92	6.00	11.00
Mixup	2.23	**7.93**	7.22	**6.58**
RegMixup (**Ours**)	**0.67**	8.36	**3.02**	7.03
SNGP	1.51	11.33	5.59	10.85
AugMix	1.89	**5.77**	4.10	9.61
DE (5×)	1.74	13.52	4.33	9.44

Methods	Clean AdaECE (↓)	CIFAR-100-C AdaECE (↓)
C100 R50		
DNN	9.47	25.17
Mixup	7.47	21.52
RegMixup (**Ours**)	3.92	13.68
DE (5×)	6.50	19.76
C100 WRN		
DNN	5.30	17.38
Mixup	3.60	16.54
RegMixup (**Ours**)	2.47	10.49
SNGP	5.65	10.89
AugMix	5.23	13.67
DE (5×)	3.92	13.47

Table 10: CIFAR calibration performance (%) without temperature scaling

	IND	**Covariate Shift**			
	ImageNet-1K (Test) AdaECE (↓)	**ImageNet-R AdaECE (↓)**	**ImageNet-A AdaECE (↓)**	**ImageNet-V2 AdaECE (↓)**	**ImageNet-Sk AdaECE (↓)**
DNN	4.90	20.48	52.30	9.58	22.94
Mixup	**2.28**	**14.70**	47.41	6.46	**18.26**
RegMixup (**our**)	3.06	17.42	**45.65**	7.34	20.85
AugMix	4.28	19.13	51.35	**3.94**	21.25
DE (5×)	3.61	17.32	51.64	7.94	19.35

Table 11: ImageNet calibration performance (%) without temperature scaling.

C Calibration Metrics without Temperature Scaling

For completeness, we report the calibration metrics over all the methods and considered datasets without the temperature scaling [Guo et al., 2017] in Tables 10 and 11. Details about the cross-validation procedure used when temperature scaling is applied is provided in Appendix A.

D Bayesian at Test Time: Last Layer Laplace Approximation

A structural problem of using MLE logistic regression is that the produced uncertainties depend on the decision boundary. On the other hand, replacing the MLE logistic regression with a Bayesian logistic regression and estimating the predictive posterior employing a Laplace approximation allows to produce better uncertainties [Kristiadi et al., 2020]. However, a Bayesian training either requires a modification in the architecture [Liu et al., 2020a] or makes the inference procedure very expensive [Kingma et al., 2015, Gal and Ghahramani, 2016]. Since the objective is to utilize the standard MLE training of neural networks, the idea of Kronecker-Factored Last Layer Laplace Approximation [Kristiadi et al., 2020] is making the network **Bayesian at test time** with almost no additional cost.

Let $\mathbf{w}$ be the parameters of the of the last layer of a neural network, then we seek to obtain the posterior only over $\mathbf{w}$. Let $p(\mathbf{w}|\mathbf{x})$ be the posterior, then the predictive distribution can be written as:

$$p(y = k|\mathbf{x}, \mathcal{D}) = \int \texttt{softmax}(\mathbf{s}_k) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}, \tag{4}$$

where, $\mathbf{s}$ is the logit vector and $\texttt{softmax}(\mathbf{s}_k)$ is the k-th index of the $\texttt{softmax}$ output of the network.

The Laplace approximation assumes that the posterior $p(\mathbf{s}|\mathcal{D}) \sim \mathcal{N}(\mathbf{s}|\mu, \Sigma)$, where μ is a mode of the posterior $p(\mathbf{w}|\mathcal{D})$ (found via standard optimization algorithms for NNs) and Σ is the inverse of the Hessian $\mathbf{H}^{-1} = -(\nabla^2 \log p(\mathbf{w}|\mathcal{D})|_\mu)^{-1}$. For the formulations and definitions, including the variants with the terms associated to the bias, we refer to [Kristiadi et al., 2020].

For our experiments, we obtain Σ using the Kronecker-factored (KF) approximation [Ritter et al., 2018]. Broadly speaking, the KF approximation allows to reduce the computational complexity of

computing the Hessian by factorizing the inverse of the Hessian as $\mathbf{H}^{-1} \approx \mathbf{V}^{-1} \otimes \mathbf{U}^{-1}$, then the covariance of the posterior evaluated at a point $\mathbf{x}$ takes following form $\Sigma = (\phi(\mathbf{x})^T \mathbf{V} \phi(\mathbf{x}))\mathbf{U}$. This procedure can be easily implemented using the Backpack library [Dangel et al., 2020] to compute $\mathbf{V}$ and $\mathbf{U}$ by performing a single pass over the training set after the end of the training, as detailed in the Appendix of [Kristiadi et al., 2020] and clearly exemplified in the code-base of [Hobbhahn et al., 2021].

Let Σ_k be the covariance matrix of the posterior over the last linear layer parameters for the k-th class obtained using the Laplace approximation around μ, then, given an input $\mathbf{x}$, we obtain $\sigma_k = \phi(\mathbf{x})^\top \Sigma_k \phi(\mathbf{x})$ representing the variance of k-th logit $\mathbf{s}_k$. Once we obtain the covariance matrix, the Monte Carlo approximation of the predictive distribution (equation (4)) is obtained as:

$$\tilde{p} = \frac{1}{m} \sum_{i=1}^{m} \texttt{softmax}(\mathbf{s}(i)), \tag{5}$$

where, m logit vectors $\mathbf{s}(i)$ are sampled from a distribution with mean $\mathbf{s}$ and a covariance matrix (depending on the approximation used). Lu et. al [Lu et al., 2020] showed that similar performance can be achieved via the mean-field approximation which provides an approximate closed form solution of the integration in equation (4) involving the re-scaling of the logits and then taking the softmax of the re-scaled logit. The re-scaling is defined as follows:

$$\tilde{\mathbf{s}}_k = \frac{\mathbf{s}_k}{\sqrt{1 + \lambda \sigma_k^2}} \tag{6}$$

Note, the scaling of the k-th logit depends on its variance (obtained using the Laplace approximation) and a hyperparameter λ. This approximation is efficient in the sense that it does not require multiple samples as required in the MC approximation (which can become expensive as the number of classes and samples grow). In our experiments, we use the MC approximation, since we could not find an obvious way to fine-tune λ. Additionally, we observe that the mean-field approximation imposes a trade-off between calibration and OOD detection performance. Increasing λ, indeed, flattens the softmax distribution and improves OOD detection scores; although, as a consequence, harms calibration by making the network underconfident.

E Additional Insights: RegMixup encourages compact and separated clusters

Here we provide additional experiments to show that RegMixup encourages more compact and separated clusters in the feature space We use the well known Fisher criterion [Bishop, 2006, Chapter 4] to quantify the compactness and separatedness of the feature clusters.

Fisher Criterion: 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 within-class covariance is small and between-class covariance is high, thus, *a high value of α is desirable*. In Figure 4, 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. However, *RegMixup consistently provides the best α* in most cases.

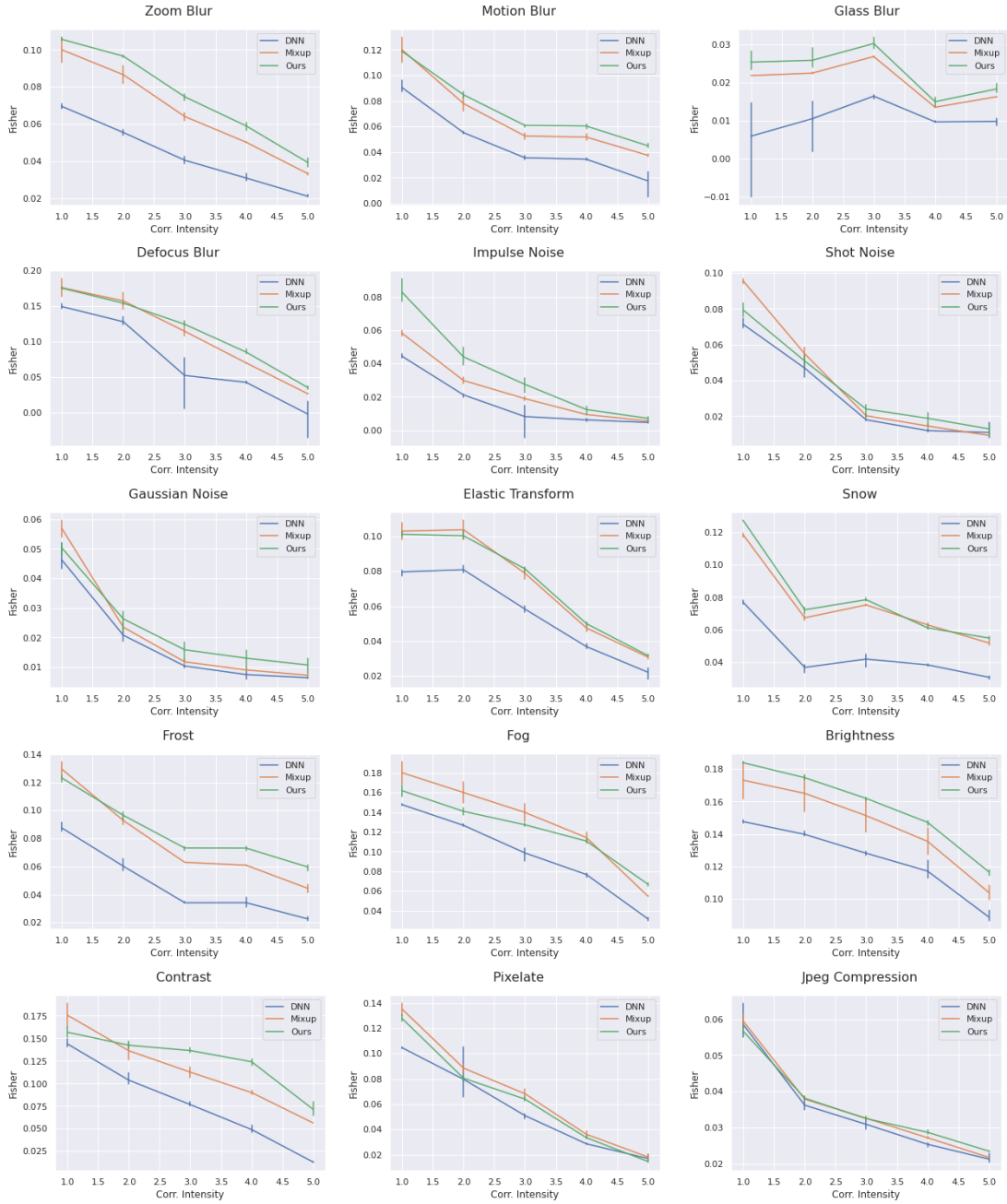

Figure 4: Fisher criterion for all the corruptions and intensity values of CIFAR-10-C (WRN28-10).

	WRN28-10		RN50	
	CIFAR10 (Val)	CIFAR100 (Val)	CIFAR10 (Val)	CIFAR100 (Val)
α	Accuracy			
0.1	96.06	81.04	95.35	79.60
0.2	96.46	80.91	95.21	80.11
0.3	96.77	81.06	95.36	80.31
0.4	96.71	81.01	95.26	78.93
0.5	96.70	80.99	95.28	78.91
1	96.74	80.66	94.96	78.79
5	96.62	79.84	94.98	77.74
10	96.54	79.24	94.94	75.76
20	96.26	78.40	95.16	75.56

	RN50 ImageNet
α	Accuracy
0.1	77.10
0.2	77.02
1	76.19
10	72.17
20	71.51

Table 12: Mixup hyperparameter cross-validation: Accuracy (%) of WideResNet28-10 and ResNet50 on validation split of C10 and C100 for varying α values (table on the left), and of ResNet50 on ImageNet (table on the right).

	DenseNet-121		PyramidNet-200	
	CIFAR10 (Val)	CIFAR100 (Val)	CIFAR10 (Val)	CIFAR100 (Val)
α	Accuracy			
0.1	95.89	80.54	96.71	82.34
0.2	96.10	80.80	96.70	82.17
0.3	96.21	80.80	96.67	81.70
0.4	96.06	79.71	96.79	82.62
0.5	95.98	80.17	96.92	81.90
1	96.07	79.08	96.89	81.80
10	95.93	75.76	96.69	79.50
20	95.74	76.03	96.60	78.75

Table 13: Mixup hyperparameter cross-validation: Accuracy (%) of DenseNet-121 and PyramidNet-200.

F ECE results

In this section we report the Expected Calibration Error (ECE) values. While ECE is more popular than AdaECE, the latter uses an adaptive binning scheme that better accounts for the bias introduced by the fact neural networks tend to concentrate most samples in high-confidence bins [Mukhoti et al., 2020].

	WRN28-10						RN50					
	CIFAR10 (Val)			CIFAR100 (Val)			CIFAR10 (Val)			CIFAR100 (Val)		
	Accuracy											
α/η	0.1	1	2	0.1	1	2	0.1	1	2	0.1	1	2
0.1	96.12	96.02	96.38	80.08	80.86	80.96	94.32	95.02	94.48	78.56	79.84	77.92
0.2	96.14	96.00	96.82	80.60	81.96	81.22	94.84	95.44	95.26	78.48	79.18	78.12
0.3	95.82	96.68	96.34	80.92	81.82	80.94	94.62	95.46	95.48	78.38	79.74	78.21
0.4	96.28	96.58	96.48	80.82	81.62	81.00	94.92	95.74	95.46	78.32	79.85	78.17
0.5	96.08	96.88	96.44	81.00	81.20	81.38	94.72	96.04	95.24	78,56	79.34	78.29
1	96.36	97.00	96.96	81.58	81.72	80.68	95.36	95.98	96.00	79.14	79.88	78.62
5	96.50	97.14	97.22	82.00	81.94	81.04	95.86	96.26	96.04	79.42	80.13	78.30
10	96.54	97.16	97.28	80.98	82.51	80.60	95.28	95.58	96.32	80.36	80.65	79.16
15	96.65	97.27	97.18	81.38	82.16	80.59	95.15	96.10	96.34	79.72	80.32	78.80
20	96.72	97.32	97.16	81.58	82.27	80.62	95.46	96.50	96.38	79.68	80.18	78.76
30	96.74	97.28	97.28	82.48	81.94	80.40	95.66	96.46	95.90	79.39	79.78	79.22

Table 14: RegMixup hyperparameter cross-validation: Accuracy (%) of WideResNet28-10 and ResNet50 on validation split of C10 and C100 for varying α and η values.

Methods	IND			
	WRN28-10		RN50	
	C10 (Test)	C100 (Test)	C10 (Test)	C100 (Test)
	ECE (↓)		ECE (↓)	
DNN	1.26	3.88	1.38	3.05
Mixup	0.94	1.16	**0.59**	7.49
RegMixup (**our**)	**0.62**	**1.65**	0.62	**1.51**
SNGP	0.84	1.95	-	-
Augmix	1.67	5.54	-	-
DE (5×)	0.81	3.31	1.27	3.15

Table 15: CIFAR IND calibration performance (%).

	IND	Covariate Shift			
	ImageNet-1K (Test)	ImageNet-R	ImageNet-A	ImageNet-V2	ImageNet-Sk
	ECE (↓)	ECE (↓)	ECE (↓)	ECE (↓)	ECE (↓)
DNN	1.85	13.52	44.91	4.17	14.51
Mixup	**1.34**	12.13	44.65	4.34	15.32
RegMixup (**our**)	1.38	13.32	**41.23**	**3.41**	15.37
AugMix	2.08	**11.31**	42.85	3.97	**14.32**
DE (5×)	1.39	13.59	42.92	4.07	17.34

Table 16: ImageNet calibration performance (%).

Methods	Covariate Shift							
	WRN28-10				R50			
	C10-C	C10.1	C10.2	C100-C	C10-C	C10.1	C10.2	C100-C
	ECE (↓)				ECE (↓)			
DNN	12.64	4.29	9.03	9.96	12.31	4.53	8.97	19.80
Mixup	7.54	4.28	7.40	10.32	**10.23**	5.59	10.59	13.57
RegMixup (**our**)	9.56	**2.87**	**6.92**	**7.98**	12.61	**3.25**	**6.73**	**12.87**
SNGP	11.33	4.32	8.68	10.45	-	-	-	-
AugMix	**4.78**	3.55	8.42	12.19	-	-	-	-
DE (5 ×)	10.11	2.98	7.53	12.43	13.12	4.32	7.10	12.53

Table 17: CIFAR CS calibration performance (%).

G Using other vicinal distributions as regularizers

In this Section we perform a preliminary analysis about using other techniques derived from Mixup as regularizers. In particular, we consider the following methods:

- CutMix Yun et al. [2019]: instead of performing the convex combination of two samples, cuts and pastes a patch of an image into another, and the area of the patch is proportional to the interpolation factor λ. We set the CutMix probability to $p = 1$ and cross-validate $\alpha \in \{0.1, 0.2, 0.3, 1, 10, 20\}$,
- RegCutMix: we replace Mixup with CutMix in our formulation. Same hyperparameters as above.
- Mixup + CutMix: we call this way the Transformers-inspired training procedure that at each iteration randomly decides (with equal probability) whether to apply Mixup or CutMix. For both Mixup and CutMix we cross-validate $\alpha \in \{0.1, 0.3, 1, 10\}$.
- RegMixup + CutMix: as above, but using Mixup as a regularizer of the cross-entropy when Mixup is selected. Hyperparameters as above, $\eta \in \{0.1, 1, 3\}$

Methods	I.I.D. C10	Covariate Shift C10-C	C10.2	C10.1	O.O.D. C100	SVHN	T-ImageNet
	Accuracy (↑)				AUROC (↑)		
WRN28-10							
Regmixup	**97.46**	**83.13**	88.05	92.79	89.63	**96.72**	90.19
CutMix	96.70	71.40	87.06	91.18	88.65	92.25	**91.70**
RegCutMix	96.79	71.26	86.84	92.10	89.24	89.47	**91.68**
Vit-Mixup+CutMix	97.23	77.60	87.85	92.30	89.23	83.46	90.86
ViT-RegMixup+CutMix	97.30	78.67	88.42	93.15	89.03	91.62	89.41
ViT-RegMixup+RegCutMix	97.47	79.10	**88.79**	**93.35**	**89.93**	94.18	90.39
R50							
RegMixup	96.71	**81.18**	86.72	**91.58**	**89.63**	**95.39**	**90.04**
CutMix	96.27	70.63	85.35	90.35	85.88	83.03	87.43
RegCutMix	95.85	70.22	84.64	90.31	85.10	84.76	87.36
ViT-Mixup+CutMix	**96.87**	76.36	**87.05**	90.75	89.19	93.39	89.76
ViT-RegMixup+CutMix	96.60	77.21	86.30	90.97	84.22	87.90	85.87
ViT-RegMixup+RegCutMix	96.47	75.36	86.01	91.28	86.36	87.21	88.42

Table 18: Accuracy and out-of-distribution detection performance (%) using other VRM techniques as regularisers for ResNet50 and WideResNet28-10 on CIFAR-10.

Methods	I.I.D. C100	Covariate Shift C100-C	O.O.D. C10	SVHN	T-ImageNet
	Accuracy (↑)		AUROC (↑)		
WRN28-10					
RegMixup (Ours)	83.25	**59.44**	**81.27**	89.32	**83.13**
CutMix	81.73	46.64	79.06	85.22	81.07
RegCutMix	82.30	47.37	80.99	84.42	81.85
ViT-Mixup+CutMix	**84.05**	54.94	80.81	85.36	81.32
ViT-RegMixup+CutMix	83.74	55.12	79.92	86.44	82.28
ViT-RegMixup+RegCutMix	83.92	55.50	80.75	**90.23**	83.07
R50					
RegMixup	81.52	**57.64**	79.44	**88.66**	82.56
CutMix	80.21	45.23	77.78	85.39	80.33
RegCutMix	79.07	44.64	77.56	80.52	79.18
ViT-Mixup+CutMix	**82.39**	56.40	80.53	81.09	81.01
ViT-RegMixup+CutMix	81.65	52.99	79.64	86.45	81.42
ViT-RegMixup+RegCutMix	82.10	53.57	**80.66**	84.43	**83.03**

Table 19: Accuracy and out-of-distribution detection performance (%) using other VRM techniques as regularisers for ResNet50 and WideResNet28-10 on CIFAR-100.

- RegMixup + RegCutMix: as above, but also using CutMix as a regularizer of the cross-entropy when CutMix is selected. Hyperparameters as above, η is kept the same for both Mixup and CutMix.

The results are reported in Tables 18 and 19.

As it can be seen: (1) in most cases RegCutMix underperforms with respect to CutMix, (2) in most cases RegMixup outperforms CutMix and RegCutMix, (3) Mixup+CutMix represents an extremely competitive method for in-distribution accuracy, but not as competitive for distribution shift and out-of-distribution detection, (4) RegMixup+CutMix is most of the times inferior to Mixup+CutMix, (5) RegMixup+RegCutMix is an extremely competitive method in several cases. We leave to future research exploring whether and how it is possible to combine Mixup-inspired techniques as

regularizers to further improve the accuracy on i.i.d. and distribution-shifted data, and the out-of-distribution detection performance. We remark that in all these setups, we remark that RegMixup is extremely competitive and several times the best method, thus pointing out that despite it's simplicity, it's extremely effective.