Posterior Uncertainty Quantification in Neural Networks using Data Augmentation

Luhuan Wu[1]
Columbia University

Sinead A. Williamson
Apple Machine Learning Research

Abstract

In this paper, we approach the problem of uncertainty quantification in deep learning through a predictive framework, which captures uncertainty in model parameters by specifying our assumptions about the predictive distribution of unseen future data. Under this view, we show that deep ensembling (Lakshminarayanan et al., 2017) is a fundamentally mis-specified model class, since it assumes that future data are supported on existing observations only – a situation rarely encountered in practice. To address this limitation, we propose MixupMP, a method that constructs a more realistic predictive distribution using popular data augmentation techniques. MixupMP operates as a drop-in replacement for deep ensembles, where each ensemble member is trained on a random simulation from this predictive distribution. Grounded in the recently-proposed framework of Martingale posteriors (Fong et al., 2023), MixupMP returns samples from an implicitly defined Bayesian posterior. Our empirical analysis showcases that MixupMP achieves superior predictive performance and uncertainty quantification on various image classification datasets, when compared with existing Bayesian and non-Bayesian approaches.

[1]Work done during an internship at Apple Machine Learning Research.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION

Reliable uncertainty quantification and robust predictive performance are crucial to the deployment of deep learning models in safety-critical applications, e.g. medical diagnosis (Filos et al., 2019), autonomous driving (Feng et al., 2018) and detection of adversarial examples (Smith and Gal, 2018). Bayesian neural networks (BNNs, MacKay, 1992; Neal, 2012) have been viewed as a principled approach to achieve this goal. BNNs elicit a prior distribution over neural network parameters θ. Given n observations $z_{1:n}$, we estimate a posterior distribution over θ, which in turn induces uncertainty in downstream predictions. However, exact posterior inference for θ in BNNs is typically intractable, requiring approximations (Blundell et al., 2015; Daxberger et al., 2021). Moreover, it can be challenging to design meaningful priors over the neural network parameter space (Rudner et al., 2022). In contrast, "non-Bayesian" approaches like deep ensembles (DE, Lakshminarayanan et al., 2017) have demonstrated promising performance in both uncertainty quantification and prediction accuracy, while maintaining simplicity in training and inference.

In this work, we take a data-driven view of uncertainty quantification and posterior prediction for deep learning models by leveraging the assumptions on the future data distribution. We build on the recent idea of Martingale posteriors (MPs, Fong et al., 2023), which transfers the prior uncertainty about parameters θ to a representation of uncertainty about future data. Concretely, MP specifies a *predictive* distribution $\mathbb{P}_\infty(z_{n+1:\infty}|z_{1:n})$ over future data $z_{n+1:\infty}$ given the observed $z_{1:n}$. The parameters fitted on a random realization from $\mathbb{P}_\infty$ can be viewed as a posterior sample under some implicit prior (Doob, 1949). By fitting separate models on different random realizations, we obtain an approximate posterior distribution of parameters θ. For example, the Bayesian bootstrap (BB, Rubin, 1981) is an instance of MP where each training dataset is drawn from a random distribution supported on existing observations. Unlike the explicit prior over θ required for BNNs (which can be hard to

specify), practitioners can leverage domain knowledge about data to specify the predictive distribution $\mathbb{P}_\infty$.

While MP offers a fresh alternative to standard Bayesian inference, its application to deep learning has mostly remained unexplored. One reason for this is because deep learning is typically used for structured, high dimensional data, such as images or text. The predictive distributions $\mathbb{P}_\infty$ used in MP tend to make simple parametric assumptions about future observations that are inappropriate for structured data (Lyddon et al., 2018; Fong et al., 2019), or are not scalable to high dimensional datasets (for example, the copula-based method of Fong et al., 2023).

To fill in the gap between MPs and deep learning applications, we first show that the existing DE method can be viewed as an MP approach by establishing its functional equivalency to BB. However, the predictive distribution $\mathbb{P}_\infty$ in BB—and, by extension, DE— assumes that all future data are random repetitions of the observed data $z_{1:n}$, making them fundamentally mis-specified. While the empirical success of BB in traditional underparametrized statistical models (e.g. Neton and Raftery, 1994; Jin et al., 2001) suggests that such mis-specification may be ignored in many cases, this is *not* the case in a deep learning context: when models are over-parameterized and data is separable, BB is not sufficient to represent the uncertainty in the underlying data distribution.

To address the above limitations, we propose a new predictive distribution, $\mathbb{P}_\infty^{(\mathrm{MMP})}$, for deep learning models with a primary focus on the image modality. At a high level, $\mathbb{P}_\infty^{(\mathrm{MMP})}$ introduces uncertainty in the vicinity of observations, which increases when future data is more dissimilar from observations. To achieve this goal, we take inspiration from the Dirichlet process, which combines the empirical distribution of $z_{1:n}$ with samples from some base measure H. We replace the empirical component of Dirichlet process (which assumes exact repeats of observations $z_{1:n}$) with a version that allows for augmented repeats of $z_{1:n}$, thereby adding plausible samples that are similar to each observation. We then specify the base measure in terms of Mixup (Zhang et al., 2018), a data augmentation technique that linearly interpolates random pairs of images and their labels. Samples from $\mathbb{P}_\infty^{(\mathrm{MMP})}$ yield a diverse set of future observations with low label uncertainty near our observations $z_{1:n}$, and higher label uncertainty as we move further from $z_{1:n}$. Such behavior is aligned with previous work that suggests that it often suffices to impose uncertainty on the boundary of the training data, rather than across the entire input space (Lee et al., 2018; Hafner et al., 2020).

We show that $\mathbb{P}_\infty^{(\mathrm{MMP})}$ can be used in a simple, ensemble-like procedure which we call MixupMP, by training multiple models on different future data streams drawn from $\mathbb{P}_\infty^{(\mathrm{MMP})}$. Grounded by the MP framework, the fitted parameters of each model can be viewed as a valid posterior sample for θ. Additionally, we devise an efficient, single-model variant of MixupMP that leverages implicit ensemble techniques introduced by Gal and Ghahramani (2016).

To summarize our contributions, (1) we demonstrate that, in a deep learning context, DE is equivalent to BB, and therefore is a form of MP; however, we argue that this form of MP is mis-specified for deep learning applications; (2) we develop MixupMP, a novel MP formulation suitable for deep learning using image data; (3) we show, through empirical study, that MixupMP can outperform existing ensemble-based approaches and other approximate Bayesian methods in predictive performance and uncertainty calibration.

2 Background

Setup and notation. We focus on the supervised learning setting where we have i.i.d. samples $\{z_i = (x_i, y_i)\}_{i=1}^n$, with x_i the input and y_i the class label belonging to one of the K classes. A model then learns a parameterized distribution $p_\theta(y = k|x)$ for $k = 1 : K$ by optimizing some loss function $l_\theta(\cdot)$ over the data.

2.1 Martingale posterior distributions

The martingale posterior (MP, Fong et al., 2023) is a recently proposed uncertainty quantification technique that offers an alternative to classical Bayesian inference. Rather than specify a prior over parameters, they posit a distribution $\mathbb{P}_\infty$ over unseen future observations $z_{n+1:\infty}$, given the observed data $z_{1:n}$. The martingale posterior is then defined as

$$\pi_n(\theta) = \int \theta^*(z_{1:\infty}) d\mathbb{P}_\infty(z_{n+1:\infty}|z_{1:n}) \qquad (1)$$

where $\theta^*(z_{1:\infty})$ is the estimator of the parameters of interest θ given $z_{1:\infty}$, typically obtained by minimizing the loss $\ell_\theta(z_{1:\infty})$. If $\mathbb{P}_\infty$ is a martingale, then $\pi_n(\theta)$ converges to the Dirac measure centered at the true θ as $n \to \infty$ (up to an equivalency set). Furthermore, $\pi_n(\theta)$ can be seen as a Bayesian posterior under some (typically unknown) prior on θ, a result of Doob's Theorem (Doob, 1949; Fong et al., 2023).

The martingale requirement is satisfied if the sequence $z_{n+1:\infty}$ is conditionally identically distributed (Berti et al., 2004). A weaker requirement, that we use in this paper, is that the sequence be infinitely exchangeable — i.e., $z_i \overset{iid}{\sim} F_\infty$ for $i > n$ given some latent measure F_∞ (De Finetti, 1937). One convenient construction for F_∞ is the posterior of a Dirichlet process (DP)

Algorithm 1: Dirichlet Process-based Martingale Posterior (DP-MP)

Input: Observation $z_{1:n}$; DP base measure H and concentration parameter c, approximation value T, # of simulations B, loss $l_\theta(\cdot)$

Result: Posterior samples $\{\theta^{(b)}\}_{b=1}^B$

for $b = 1$ **to** B **do**

  Draw pseudo data $z_{n+1:n+T}^{(b)} \sim H$

  Draw weights

$$w_{1:n+T}^{(b)} \sim \text{Dirichlet}(\underbrace{1, \cdots, 1}_{\text{length } n}, \underbrace{c/T, \cdots, c/T}_{\text{length } T})$$

  Compute $\theta^{(b)} = \arg\min_\theta \sum_{i=1}^{n+T} w_i l_\theta(z_i)$

with base measure H and concentration parameter c (as proposed by Fong et al., 2019),

$$F_\infty \sim \text{DP}\left(c+n, \tfrac{cH+\sum_{i=1}^n \delta_{z_i}}{c+n}\right), \tag{2}$$

where $F_\infty := \sum_{i=1}^n w_i \delta_{z_i} + \sum_{i=n+1}^\infty w_i \delta_{\phi_i}$, with $\phi_i \sim H$. For the first n weights (corresponding to the original $z_{1:n}$), we have $\mathbb{E}[w_i] = 1/(n+c)$; for the tail we have $\mathbb{E}\left[\sum_{j=n+1}^\infty w_j\right] = c/(n+c)$. This formulation captures the idea that future data are likely to be a combination of repeats of empirical observations $z_{1:n}$, and samples from a distribution H that captures prior beliefs or some data-driven centering model, with $r = c/n$ capturing the ratio of the two.

A sample $F_\infty^{(b)} = \sum_{i=1}^n w_i^{(b)} \delta_{z_i} + \sum_{i=n+1}^\infty w_i^{(b)} \delta_{\phi_i^{(b)}}$ from Equation (2) describes the empirical distribution of a sampled sequence $z_{1:\infty}^{(b)}$ [2]. One can further obtain a sample $\theta^{(b)}$ from the corresponding martingale posterior as $\theta^{(b)} = \arg\min_\theta \int \ell_\theta(z_{1:\infty}) dF_\infty^{(b)}(z_{1:\infty}) = \arg\min_\theta \sum_{i=1}^n w_i^{(b)} \delta_{z_i} + \sum_{i=n+1}^\infty w_i^{(b)} \delta_{\phi_i^{(b)}}$. In practice, we can approximate the infinite $F_\infty^{(b)}$ with a finite measure. The full procedure to to generate B posterior samples from the DP-based martingale posterior (DP-MP) is given in Algorithm 1.

Bayesian bootstrap. If $c = 0$ in the above construction, we recover the Bayesian bootstrap (BB, Rubin, 1981), where $F_\infty^{(b)} = \sum_{i=1}^n w_i^{(b)} \delta_{z_i}$, with

$$(w_1^{(b)}, \ldots, w_n^{(b)}) \sim \text{Dirichlet}(1, \ldots, 1). \tag{3}$$

We then optimize the weighted loss to obtain

$$\theta^{(b)} = \arg\min_\theta \sum_{i=1}^n w_i^{(b)} l_\theta(z_i). \tag{4}$$

[2] Since n is finite, the distribution of $z_{1:\infty}$ is almost surely equal to the distribution of $z_{n+1:\infty}$.

BB is a special case of the MP that assumes all future observations are repeats of the n original observations.

2.2 Ensemble methods in deep learning

Deep ensembles. A deep ensemble (DE, Lakshminarayanan et al., 2017) is a collection of B neural networks, typically with the same architecture, trained from different random initializations of θ. Specifically, each network obtains its set of parameters $\{\theta^{(b)}\}_{b=1}^B$ by minimizing the empirical loss,

$$\theta^{(b)} = \arg\min_\theta \sum_{i=1}^n l_\theta(z_i). \tag{5}$$

The prediction is then made by combining outputs from individual networks, e.g. averaging their classification logits. DE has been shown to perform similarly to or better than alternative Bayesian neural networks in uncertainty quantification, predictive accuracy and robustness to distribution shifts (Ovadia et al., 2019).

Monte Carlo dropout. Monte Carlo dropout (MC Dropout, Gal and Ghahramani, 2016) trains a single neural network using the dropout technique (Srivastava et al., 2014). During inference, MC Dropout simulates an ensemble of models by randomly activating dropout in separate forward passes. This procedure can be interpreted either as an approximation to DE (Hara et al., 2016) or as a variational inference method for BNN (Gal and Ghahramani, 2016).

2.3 Mixup

Mixup (Zhang et al., 2018) is a data augmentation technique that samples an augmentation $z' = (x', y')$ as a convex combination of two random observations $z_i = (x_i, y_i)$ and $z_j = (x_j, y_j)$ from $z_{1:n}$,[3]

$$\left.\begin{array}{ll} \lambda & \sim \text{Beta}(\alpha, \alpha) \\ z_i, z_j & \overset{iid}{\sim} \{z_1, \ldots, z_n\} \\ x' & = \lambda x_i + (1-\lambda) x_j \\ y' & = \lambda y_i + (1-\lambda) y_j \end{array}\right\} \quad z' \sim \text{Mixup}(z_{1:n}; \alpha) \tag{6}$$

for some $\alpha > 0$. Directly training a model on these augmented examples has been found to improve test set accuracy and calibration performance (Zhang et al., 2018; Thulasidasan et al., 2019). When used in an ensemble, however, Mixup has been shown to have worse uncertainty calibration performance than the single-model Mixup or deep ensemble trained on original data (Rahaman and Thiery, 2021; Wen et al., 2021).

[3] In this paper, we assume a separate λ is generated for each sample $z' \sim \text{Mixup}(z_{1:n}; \alpha)$.

3 An equivalency between DE and BB

In this section, we show that DE can be viewed as an instance of martingale posteriors, through establishing its equivalency to a finite approximation to the BB in many practical deep learning scenarios; furthermore, we reveal that the predictive distribution underlying such martingale posteriors is mis-specified, motivating the main methodology in the next section.

We first note that both BB and DE are carrying out weighted empirical loss minimization, albeit with different sets of weights (Equations (4) and (5)). Recent theoretical work has shown that, if training data is separable and under certain conditions on the neural network, loss, and optimizer, a neural network will converge to the L2 max margin solution (Lyu and Li, 2020; Wei et al., 2019; Nacson et al., 2019; Chizat and Bach, 2020). Further, Xu et al. (2021) have shown that this margin is invariant to replacing the empirical loss $\sum_i \ell_\theta(z_i)$ with a weighted loss $\sum_i w_i \ell_\theta(z_i)$, where the w_i are bounded weight values.

Combining these observations leads to the following proposition:

Proposition 1. *(Informal) If a dataset $z_{1:n}$ is separable under a given homogeneous neural network with parameters θ, trained via stochastic gradient descent using an exponentially tailed loss (e.g. cross-entropy) with weak regularization, then any posterior sample of θ obtained via an appropriately stabilized version of BB could also have been obtained via DE, and vice versa.*

See the formal statement, assumptions and proof, plus definition of stabilized BB, in Appendix A.

This result extends previous empirical work (Nixon et al., 2020) that shows the frequentist bootstrap (FB, Efron, 1992) underperforms DE in deep learning applications. The authors attributed FB's underperformance to the fact that each bootstrap sample only contains 63.7% of the unique observations from the training data, undermining the generalization power of the learned model.

In BB, since all observations are represented in every bootstrap sample, one might expect to avoid the pitfall of FB, and even outperform DE. However, Proposition 1 suggests that it is not the case, as BB is functionally equivalent to DE. In fact, Section 6.1 shows that there is little difference in their empirical performance.

Mis-specification of BB and DE. The success of BB relies on the assumption that the predictive distribution of future observations can be well approximated by the empirical distribution. Clearly this assumption is not true: the test set will not include only repeats of the training data. In an underparametrized model, this mis-specification can typically be ignored; since a perfect training accuracy is rarely achieved, differently weighted training set causes the model to focus on different regions, effectively capturing some uncertainty in the input space.

Conversely, BB's assumptions are fundamentally misspecified in the deep learning context. Overparameterized deep neural networks perfectly fits all training observations, and hence different weights do not impact their convergent solutions (in the asymptotic regime). However, realistic test cases involve novel observations, and including these observations to the training set which *will* impact the convergent solutions of models. For this reason, we argue that BB — and by extension of Proposition 1, DE — correspond to mis-specified versions of martingale posteriors.

4 Mixup Martingale posteriors: Incorporating prior knowledge about the distribution of interest

Instead of concentrating on existing observations as in BB or DE, a better choice of the predictive distribution would allow for plausible additional observations that reflect our beliefs and uncertainty about future data. The DP-MP approach discussed in Section 2 adds in such additional observations. For low-dimensional or unstructured data we can obtain plausible new samples using relatively simple base measures (see Appendix B for examples). However, such measures do not translate well to highly structured data such as images: sampling from a moment-matched Gaussian on pixel space will not yield realistic images, for example.

Instead, we propose MixupMP, an MP method that models the predictive uncertainty with data augmentation techniques. We focus on image data, for which data augmentation has proved highly successful (Shorten and Khoshgoftaar, 2019), although the general ideas could be extended to alternative modalities (e.g., Feng et al., 2021; Meng et al., 2021).

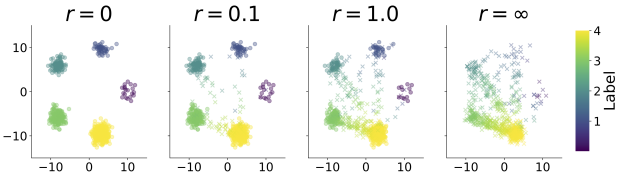
We start from the DP-MP approach, which draws samples of future data that are either copies of previous observations, or "new" observations from a base measure H. In practice we do not expect future data to be exact copies of $z_{1:n}$; instead we assume variations of original data obtained by standard randomized label-preserving data augmentation techniques such as random cropping. Hence we replace the point masses δ_{z_i} in Equation (2) with a distribution $H_{z_i}^{(\text{aug})}(\cdot)$, such that samples from $H_{z_i}^{(\text{aug})}$ are generated by (i) sampling a random augmentation from some set $\mathcal{T}^{\text{aug}}$, (ii) applying that augmentation to x_i, and (iii) combining with

Algorithm 2: MixupMP using mini-batches

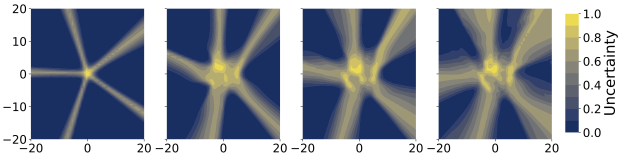
Data: Data $\{z_i = (x_i, y_i)\}_{i=1}^n$, concentration ratio $r := c/n$, Mixup parameter α, minibatch size n_{mb}, pseudosample batch size t_{mb}, # of simulations B, loss $l_\theta(\cdot)$, data loader $d_{n_{mb}}(z_{1:n})$

Result: Posterior samples $\{\theta^{(b)}\}_{b=1}^B$

for $b = 1$ **to** B **do**
  Randomly initialize $\theta^{(b)}$
  while *not converged* **do**
    for $z_{1:n_{mb}}^{(mb)}$ ***in*** $d_{n_{mb}}(z_{1:n})$ **do**
      for $i = 1$ **to** n_{mb} **do**
        Sample $\tilde{z}_i \sim H^{(\text{aug})}_{z_i^{(mb)}}$
      for $i = n_{mb} + 1$ **to** $n_{mb} + t_{mb}$ **do**
        Sample $\tilde{z}_i \sim H^{(\text{Mixup})}_{\tilde{z}_{1:n_{mb};\alpha}}$
      Update $\theta^{(b)}$ by gradient descent on $\sum_{i=1}^{n_{mb}} \ell_{\theta^{(b)}}(\tilde{z}_i) + r\frac{n_{mb}}{t_{mb}} \sum_{i=n_{mb}+1}^{n_{mb}+t_{mb}} \ell_{\theta^{(b)}}(\tilde{z}_i)$



(a) $F_\infty^{(\text{MMP})}$ sample from MixupMP with varying ratio $r := c/n$ and $H_x^{(\text{aug})} := \delta_x$ and $\alpha = 1.0$. Dots represent samples from observations and crosses are samples from the base measure. The label space is extended to interval $[0, K-1]$.



(b) The corresponding predictive uncertainty landscape.

Figure 1: Illustration of MixupMP on synthetic classification task ($K = 5$) with $\alpha = 1.0$. As r increases, $F_\infty^{(\text{MMP})}$ puts more uncertainty on the space between observations, inducing higher predictive uncertainty.

the original y_i.

Second, we specify a data-driven base measure $H^{(\text{Mixup})}_{z_{1:n};\alpha}$. We sample from $H^{(\text{Mixup})}_{z_{1:n};\alpha}$ by first applying random augmentations to $x_{1:n}$ to get augmented inputs $\tilde{x}_{1:n}$, then sampling an observation z' according to Mixup$(\tilde{z}_{1:n}, \alpha)$, where $\tilde{z}_i = (\tilde{x}_i, \tilde{y}_i)$ (Equation (6)).

Finally, we make a practical choice to replace the weights $\{w_i\}_{i=1}^\infty$ sampled from the DP with their expectations, simplifying implementation. This choice is further motivated by that randomized weights make little difference in separable settings (see Section 3 and Section 6.1). The resulting distribution takes the form

$$F_\infty^{(\text{MMP})}(\cdot) \propto \sum_{i=1}^n H_{z_i}^{(\text{aug})}(\cdot) + rH^{(\text{Mixup})}_{z_{1:n};\alpha}(\cdot), \qquad (7)$$

where $r := c/n$ is the concentration ratio parameter for some $c \geq 0$. $\mathbb{P}_\infty^{(\text{MMP})}$ is then the future predictive distribution implied by i.i.d. sampling from $F_\infty^{(\text{MMP})}$.

We obtain posterior samples of θ by repeatedly sampling a sequence of observations from $F_\infty^{(\text{MMP})}$ and then minimizing the loss l_θ with respect to this sequence. We note that this sequence is exchangeable by construction, and thus this procedure generates samples from a well-defined martingale posterior. We call this approach MixupMP, and summarize a practical procedure in Algorithm 2.

In Algorithm 2, we split the training sequence into mini-batches, allowing it to effectively work with a sequence of unbounded length. Additionally, we use a fixed dataloader (permuted in the beginning of every training epoch) that cycles through the data rather than randomly sampling. While the resulting samples are no longer exact samples from $F_\infty^{(\text{MMP})}$ (Equation (7)), we expect the practical impact would be minimal.

Relationship to other methods. If the concentration ratio $r = 0$ in Equation (7), MixupMP reduces to DE with standard augmentation. If, in addition, $H_z^{(\text{aug})} := \delta_z$, we recover DE without augmentation. If $r = \infty$ in Equation (7) and $B = 1$ in Algorithm 2, we recover the original Mixup algorithm (Zhang et al., 2018). And if $r = \infty$ with $B > 1$, we recover Mixup Ensemble, as explored by Rahaman and Thiery (2021) and Wen et al. (2021).

Illustration of the predictive distribution. We illustrate the effect of the concentration ratio r in a synthetic 5-class dataset in Figure 1. When $r = \infty$, all samples from $F_\infty^{(\text{MMP})}$ are interpolations between observations, as shown in the RHS of Figure 1a. However, as we decrease r, we see an increasing proportion of samples that are close to the original empirical distribution. This behavior avoids over-smoothing and ensures we are training on sufficient data points that are close to actual observations, while still populating regions between clusters. We can think of r as describing the extent to which we believe future observations will look like our training data (small r) vs the Mixup base measure (large r).

Figure 1b shows the corresponding uncertainty estimates obtained using MixupMP (see Appendix B for

details). The DE solution ($r = 0$) provides low uncertainty across most of the space except around the classification boundary. As we increase r, we see increase uncertainty in the regions between the original 5 clusters of observations. And Mixup Ensemble ($r = \infty$) provides excessive uncertainty even in the region when observations are present.

The choice of the Mixup parameter α impacts how close the novel samples from the base measure H^{Mixup} are to the data. For small values, we tend to see samples close to original observations; for larger values, we see more space-filling behavior. We repeat Figure 1 in Appendix B using more values of α.

4.1 Approximate MixupMP

We propose an efficient approximation to MixupMP using MC dropout, where a single neural network is trained with a positive dropout rate using Algorithm 2 (i.e. $B = 1$), and dropout is used at test time to generate multiple samples. Since dropout is randomly activated in each training iteration, this procedure implicitly trains an ensemble of models over different draws of data from $F_\infty^{(\text{MMP})}$ (see Section 2). We refer to this approximation as MixupMP-MC.

5 Related works

Several works have applied bootstrapping-type methods to deep learning, some of which fall under the definition of martingale posteriors. Osband et al. (2016) use a frequentist bootstrap to perform inference in deep Q-networks (DQN), improving performance over a single DQN model. Shin et al. (2021) use a hypernetwork approach to approximately implement BB in neural networks. Newton et al. (2021) and Osband et al. (2018) modify the BB to incorporate a randomized prior term in the loss function. Lee et al. (2023) specifically follow a MP approach, specifying $\mathbb{P}_\infty$ using an exchangeable generative neural network to quantify uncertainty for neural processes (Garnelo et al., 2018); however their approach is specific to neural processes (Garnelo et al., 2018) rather than general neural networks. By comparison, MixupMP is data-driven and applies to general architectures.

More broadly, several approximate inference methods for BNNs have been proposed. In addition to DE and MC Dropout (discussed in Section 2), options include Monte Carlo methods (MacKay, 1992; Chen et al., 2014; Zhang et al., 2020); variational inference (Graves, 2011; Blundell et al., 2015); Laplace approximations (Daxberger et al., 2021); and expectation propogation (Hernández-Lobato and Adams, 2015). With the exception of DE and Monte Carlo methods, these approaches only explore a single mode of the posterior. Sampling from multiple modes has been found to improve the quality of posterior estimates (Wilson and Izmailov, 2020), and the posterior estimate obtained using DE has been shown to be closer to a "gold standard" Monte Carlo estimate than variational approaches (Izmailov et al., 2021).

When $r = \infty$, MixupMP corresponds to a ensemble of models trained using Mixup (which we refer to as Mixup Ensemble). This setting has been shown to yield underconfident predictions (Wen et al., 2021; Rahaman and Thiery, 2021). Calibration-adjusted Mixup (CAMixup, Wen et al., 2021) is a modification of Mixup Ensemble that aims to reduce this underconfidence by only applying mixup to classes where the model is already over-confident (as assessed on a validation set after each training epoch). Consequently, all members of CAMixup need to be trained simultaneously. In contrast, our approach allows individual training of each model, alleviating memory constraints.

6 Experiments

In this section, we provide empirical results supporting the equivalency of BB and DE, and empirically evaluate the performance of MixupMP. For code, see `https://github.com/apple/ml-MixupMP`. Experiments were carried out using Apple internal clusters.

When comparing BB and DE, we look at two datasets: MNIST (LeCun et al., 1998) and FMNIST (Xiao et al., 2017), and do not use any data augmentations. For the analysis of MixupMP, we look at three datasets: CIFAR10, CIFAR100 (Krizhevsky et al., 2009), and FMNIST. For the CIFAR datasets, we specify $H^{(\text{aug})}$ using the augmentations `RandomResizeCrop` and `RandomHorizontalFlip` in PyTorch (Paszke et al., 2019); we also use these augmentations in competing methods. For FMNIST, we do not use augmentations since the images are centered and standardized. Unless otherwise stated, we use $B = 4$ ensemble members for all DE, Mixup Ensemble, BB and MixupMP experiments to estimate the posterior mean, and $B = 20$ for implicit ensemble methods based on MC Dropout, following Wen et al. (2021). We include additional implementation details in Appendix D and Appendix E.

Evaluation Metrics. We use 0-1 accuracy (ACC) and negative log likelihood (NLL) to evaluate predictive performance. For uncertainty quantification, we consider expected calibration error (ECE, Naeini et al., 2015), over-confidence error (OE), and underconfidence error (UE); see Appendix C for details. Fi-

Table 1: Comparing BB with DE. Models are either randomly initialized (same set of seeds for DE and BB), or initialized from a pretrained DE.

Dataset	Method	Init.	Acc (%)	ECE (%)
MNIST	DE	random	99.33	0.41
	BB	random	99.17	0.24
	DE	DE	99.33	0.40
	BB	DE	99.33	0.41
FMNIST	DE	random	91.52	2.32
	BB	random	91.21	2.01
	DE	DE	91.57	2.42
	BB	DE	91.55	2.37

nally, we assess the predictive entropy as a predictor for uncertainty in the event of distribution shift.

6.1 Equivalency of BB and DE

In Section 3, we argued that, on separable datasets, BB is equivalent to DE. Here, we demonstrate this claim empirically. We compare test set performance on MNIST and FMNIST under BB and DE, each with $B = 4$ simulations. In every simulation, the model is run until it achieves 100% training accuracy, plus 1000 further epochs, to ensure a separable setting is achieved. The initializations were kept fixed between the two approaches, either from a random initialization, or from a separately pretrained DE.

From Table 1, we see little difference between the two methods in terms of test accuracy and ECE when the initialization is set to a good solution provided by a separately pretrained DE. Moreover, in Appendix D, we show that the individual ensemble members within BB and DE do not significantly differ in this scenario.

When DE and BB are randomly initialized, there is a small difference between the two. We note that Proposition 1 only concerns the convergent behavior; it is possible that the weights added in BB nudge the models to different basins of attraction during early-stage training dynamics. We may be also seeing differences in convergence rate. This result supports Byrd and Lipton (2019), who find that, while the effect of importance weighting will ultimately vanish as training progresses, empirically "models with more extreme weighting converge more slowly in decision boundary". In our case, it could take BB longer to converge due to variation among the random weights, as composed to the uniform weights used by DE.

6.2 Ablation study: Impact of hyperparameters in MixupMP

The performance of MixupMP hinges on the ability of $F_\infty^{(\mathrm{MMP})}$ to capture uncertainty about future data. This ability depends on the choices of random augmentations in $H^{(\mathrm{aug})}$, the Mixup parameter α used in Mixup, and the concentration ratio r of Mixup samples to (augmented) observations.

In Figure 2, we look at the performance of MixupMP on CIFAR10, across various values of α and r. Note that when $r = 0$ we recover DE (which does not rely on α), and when $r = \infty$, the Mixup Ensemble.

For any fixed α, we observe MixupMP's predictive performance (ACC and NLL) with moderate value of r improves upon both the $r = 0$ and $r = \infty$ solutions (i.e., DE and Mixup Ensemble). For uncertainty calibration, MixupMP with $r = 0.1$ achieves slightly better ECE compared to DE ($r = 0$), but larger r can greatly inflate the ECE, especially for Mixup Ensemble ($r = \infty$). This poor calibration performance of Mixup Ensemble also corroborates previous findings in Wen et al. (2021). Looking at OE and UE, we observe that increasing r tends to reduce the model's overconfidence but boosting the under-confidence. This result is expected since we are increasing the relative importance of uncertain training examples.

On the other hand, for moderate $0 < r < \infty$, raising α generally leads to improvements in ACC and NLL, with only a marginal increase in ECE. However for Mixup Ensemble ($r = \infty$), large values of α can significantly hurt NLL and ECE, despite enhancing the accuracy. In theory, increasing α will shift the Mixup base measure to higher uncertainty region, with $\alpha \to \infty$ leading to maximum uncertainty since pairs of observations are equally mixed. Such effect of large α can be amplified with large r, explaining the uncertainty calibration behavior of Mixup Ensemble.

We conclude that the optimal value of r should be somewhere between the extremes of 0 and ∞. DE ($r = 0$) assumes that *all* future data are copies of previous observations, which leads to overconfident estimates and worse accuracy on new data points. Conversely, Mixup Ensemble ($r = \infty$) assumes that *all* future data are uncertain, leading to underconfidence despite an improved accuracy. MixupMP interpolates between these two extremes, balancing the predictive performance and uncertainty calibration. We include additional analysis on CIFAR100 and FMNIST in Appendix E where we observe similar trends.

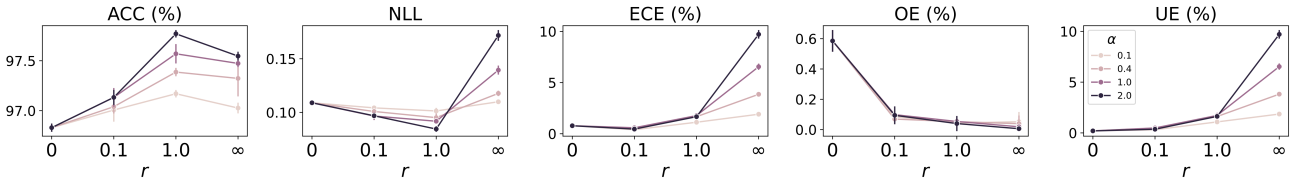


Figure 2: Impact of α and r on test set performance of MixupMP on CIFAR10. $r = 0$ corresponds to DE; $r = \infty$ corresponds to Mixup Ensemble. Results for CIFAR100 and FMNIST are included in Appendix E.

Table 2: Performance comparison on three datasets. Bolded metrics are the best (within 2 standard errors, omitted for space) in each group of {single model, explicit ensemble, implicit ensemble}; $*$ indicates the best among all methods. Single model results are averaged over 6 independent runs and ensemble results are averaged over 3 independent ensembles. We include all ablation results with standard errors in Appendix E

Dataset	CIFAR10			CIFAR100			FMNIST		
Metric	ACC (%)	NLL	ECE (%)	ACC (%)	NLL	ECE (%)	ACC (%)	NLL	ECE (%)
Single model									
NN	96.12	0.1473	2.02	80.82	0.7816	**4.78**	93.72	0.2181	2.89
Mixup (α=2.0)	**96.88**	0.1854	8.59	**81.83**	**0.7614**	8.54	**94.15**	**0.1948**	**1.46**
Laplace	96.04	**0.1344**	**0.80**	80.95	1.0107	22.96	89.67	0.4088	5.89
Explicit ensemble (B=4)									
DE	96.83	0.1090	0.78	83.28	0.6413	**3.14**	94.30	0.1768	1.36
MixupMP (r=0.1, α=2.0)	97.13	0.0969	**0.46***	84.46	0.6127	3.95	**94.75***	**0.1610***	1.27
MixupMP (r=1.0, α=2.0)	**97.77***	**0.0845***	1.67	**85.98***	**0.5548***	5.49	**94.70***	0.1662	**1.01**
Mixup Ensemble (α=2.0)	97.55	0.1717	9.71	84.48	0.6768	13.30	**94.74***	0.1776	2.47
Implicit ensemble (B=20)									
MC Dropout	96.16	0.1315	1.46	80.83	0.7451	3.69	94.41	0.1806	1.90
MixupMP-MC (r=0.1, α=2.0)	96.58	0.1145	**0.42***	82.42	0.7079	3.39	**94.72***	0.1651	1.46
MixupMP-MC (r=1.0, α=2.0)	**97.27**	**0.1005**	1.21	**83.58**	**0.6419**	4.21	**94.81***	**0.1619***	**0.97***
Mixup-MC (α=2.0)	96.80	0.2084	10.98	81.75	0.7616	10.71	94.69	0.1796	2.84
CAMixup-MC (α=2.0)	96.11	0.1365	1.21	80.19	0.7780	**2.14***	94.27	0.1818	**1.04***

6.3 Comparison with other uncertainty quantification methods

Next, we evaluate how MixupMP performs relative to standard neural networks, and to other Bayesian and ensemble-based methods. These methods include (1) *NN*, a single neural network; (2) *Mixup*, a single neural network trained with Mixup augmentation; (3) *Laplace*, a Laplace approximation to the BNN posterior (Daxberger et al., 2021); (2) *MC Dropout*; (3) *DE*; (4) *Mixup Ensemble*; (5) *CAMixup-MC*, an efficient version of CAMixup (Wen et al., 2021).[4] See Appendix E for details. Additionally, we consider *MixupMP-MC*, an efficient approximation to MixupMP as introduced in Section 4.1. Similarly we include *Mixup-MC* as an approximation to Mixup Ensemble (equivalent to MixupMP-MC with $r = \infty$). For all the methods that use Mixup augmentation, we set $\alpha = 2.0$. We refer to the class of models that form a distribution over parameters as *probabilistic* models. All above methods methods except for single NN and Mixup are probabilistic. The results are summarized in Table 2.

We first note that single NN underperforms probablistic models in almost all cases, supporting findings from earlier works (Izmailov et al., 2021; Ovadia et al., 2019). Among the probabilistic models, we find that MixupMP with $r = 1.0$ outperforms Laplace, MC Dropout, DE and Mixup Ensemble in terms of accuracy and NLL, either using explicit ensemble or implicit ensemble. MixupMP with $r = 0.1$ performs slightly worse than $r = 1.0$ on these two metrics, but is better than other approaches in ECE. In terms of uncertainty calibration, MixupMP with $r = 0.1$ is the best or close to the best among all methods.

We next examine the performance of MixupMP's more efficient variant, MixupMP-MC. While MixupMP-MC performs slightly worse than MixupMP, it outperforms MC Dropout, Laplace, and Mixup-MC on all metrics. CAMixup-MC achieves good calibration performance—its primary goal—as it has a separate validation set to adjust calibration. However, CAMixup-MC underperforms MixupMP and MixupMP-MC in terms of accuracy and NLL. These observations highlight that MixupMP-MC can serve as an efficient proxy with little performance compromise.

[4]While Wen et al. (2021) propose two other ensembling versions of CAMixup, they either impose significant memory constrainst, or require modifications to the underlying neural network.

6.4 Robustness to distribution shift

Lastly, we evaluate the generalization power and the reliability of uncertainty estimates of MixupMP in distribution shift settings. We consider the CIFAR10-C dataset which contain 19 types of corruptions to the original CIFAR10 test set, each with 5 shift intensity levels (Hendrycks and Dietterich, 2018).

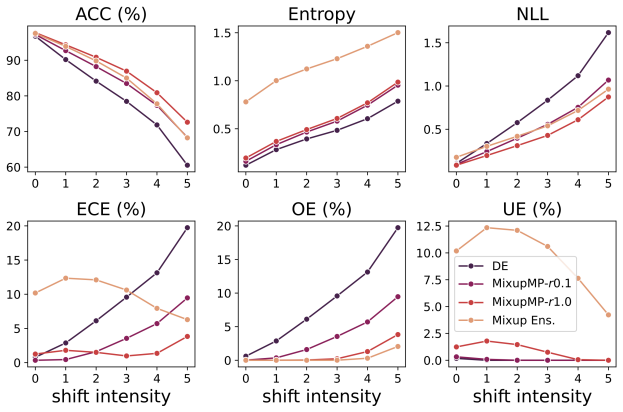


Figure 3: Performance under distribution shift using CIFAR10-C dataset. The distribution shift intensity ranges from 0 to 5, where 0 indicates no shift.

In Figure 3, we compare the performance of DE, MixupMP, and Mixup Ensemble, each with the Mixup parameter $\alpha = 2$. For all methods, as the test data shift intensity increases, both ACC and NLL deteriorate as expected; however, performance deteriorates less for MixupMP and Mixup Ensemble. There is a concurrent increase in the predictive entropy for all models, suggesting that their uncertainty estimates are informative. In particular, the higher the concentration ratio r for MixupMP, the higher the predictive entropy for the same shift intensity (recalling that DE corresponds to MixupMP with $r = 0$ and Mixup Ensemble corresponds to $r = \infty$).

Regarding calibration, in alignment with our findings in the in-distribution setting in Section 6.2, DE tends to be over-confident and Mixup Ensemble tends to be under-confident. Meanwhile, MixupMP with moderate values of r avoids both pitfalls, achieving better calibration in almost all cases.

Finally, we highlight that MixupMP with $r = 1.0$ achieves the best accuracy, NLL and ECE across all shift intensity levels in Figure 2. We include additional results on CIFAR100-C dataset (Hendrycks and Dietterich, 2018) and comparison to other methods in Appendix E. Through this comprehensive study, we show that MixupMP achieves superior performance in various distribution shift settings.

7 Discussion

In this work, we show that the posterior distribution implied by deep ensembles can be framed as a martingale posterior, but caution that when viewed in this light, it is misspecified in most settings. Instead, we propose MixupMP, a novel martingale posterior approach that uses state-of-the-art data augmentation techniques to better captures predictive uncertainty in image data, leading to improved predictive performance and uncertainty quantification. We hope that this work will spark increased interests in predictive approaches to quantify uncertainty for deep learning, and inspire future work on other structured data modalities.

References

Abe, T., Buchanan, E. K., Pleiss, G., Zemel, R., and Cunningham, J. P. (2022). Deep ensembles work, but are they necessary? In *Advances in Neural Information Processing Systems*.

Berti, P., Pratelli, L., and Rigo, P. (2004). Limit theorems for a class of identically distributed random variables. *The Annals of Probability*, 32(3):2029–2052.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International Conference on Machine Learning*.

Byrd, J. and Lipton, Z. (2019). What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*.

Chizat, L. and Bach, F. (2020). Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. (2021). Laplace redux-effortless Bayesian deep learning. In *Advances in Neural Information Processing Systems*.

De Finetti, B. (1937). La prévision: ses lois logiques, ses sources subjectives. *Annales de l'institut Henri Poincaré*, 7(1):1–68.

Doob, J. L. (1949). Application of the theory of martingales. *Le calcul des probabilites et ses applications*, pages 23–27.

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pages 569–593. Springer.

Feng, D., Rosenbaum, L., and Dietmayer, K. (2018). Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *International Conference on Intelligent Transportation Systems*.

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., and Hovy, E. (2021). A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP*.

Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., Alizadeh, M., De Kroon, A., and Gal, Y. (2019). A systematic comparison of Bayesian deep learning robustness in diabetic retinopathy tasks. In *NeurIPS Workshop on Bayesian Deep Learning*.

Fong, E., Holmes, C., and Walker, S. G. (2023). Martingale posterior distributions. *Journal of the Royal Statistical Society, Series A*, 85(5):1357–1391.

Fong, E., Lyddon, S., and Holmes, C. (2019). Scalable nonparametric sampling from multimodal posteriors with the posterior bootstrap. In *International Conference on Machine Learning*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. (2018). Neural processes. *arXiv preprint arXiv:1807.01622*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24.

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., and Davidson, J. (2020). Noise contrastive priors for functional uncertainty. In *Uncertainty in Artificial Intelligence*.

Hara, K., Saitoh, D., and Shouno, H. (2016). Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Hendrycks, D. and Dietterich, T. G. (2018). Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. (2021). What are Bayesian neural network posteriors really like? In *International Conference on Machine Learning*.

Ji, Z. and Telgarsky, M. (2020). Directional convergence and alignment in deep learning. In *Advances in Neural Information Processing Systems*.

Jin, Z., Ying, Z., and Wei, L. (2001). A simple resampling method by perturbing the minimand. *Biometrika*, 88(2):381–390.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Lee, H., Yun, E., Nam, G., Fong, E., and Lee, J. (2023). Martingale posterior neural processes. In *International Conference on Learning Representations*.

Lee, K., Lee, H., Lee, K., and Shin, J. (2018). Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*.

Lyddon, S., Walker, S., and Holmes, C. C. (2018). Nonparametric learning from Bayesian models with randomized objective functions. In *Advances in Neural Information Processing Systems*.

Lyu, K. and Li, J. (2020). Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

Meng, L., Xu, J., Tan, X., Wang, J., Qin, T., and Xu, B. (2021). Mixspeech: Data augmentation for low-resource automatic speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*.

Nacson, M. S., Lee, J., Gunasekar, S., Savarese, P. H. P., Srebro, N., and Soudry, D. (2019). Convergence of gradient descent on separable data. In *In-*

ternational Conference on Artificial Intelligence and Statistics.

Naeini, M. P., Cooper, G., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. In *AAAI Conference on Artificial Intelligence*, volume 29.

Neal, R. M. (2012). *Bayesian learning for neural networks*. Springer Science & Business Media.

Neton, M. and Raftery, A. (1994). Approximate bayesian inference by the weighted likelihood bootstrap (with discussion). *Journal of the Royal Statistical Society Series B*, 56:1–48.

Newton, M. A., Polson, N. G., and Xu, J. (2021). Weighted Bayesian bootstrap for scalable posterior distributions. *Canadian Journal of Statistics*, 49(2):421–437.

Nixon, J., Lakshminarayanan, B., and Tran, D. (2020). Why are bootstrapped deep ensembles not better? In *"I Can't Believe It's Not Better!"NeurIPS 2020 workshop*.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*.

Rahaman, R. and Thiery, A. (2021). Uncertainty quantification and deep ensembles. In *Advances in Neural Information Processing Systems*.

Rubin, D. B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, pages 130–134.

Rudner, T. G., Chen, Z., Teh, Y. W., and Gal, Y. (2022). Tractable function-space variational inference in bayesian neural networks. In *Advances in Neural Information Processing Systems*.

Shin, M., Cho, H., Min, H.-s., and Lim, S. (2021). Neural bootstrapper. In *Advances in Neural Information Processing Systems*.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48.

Smith, L. and Gal, Y. (2018). Understanding measures of uncertainty for adversarial example detection. In *Uncertainty in Artificial Intelligence*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*.

Wang, K. A., Chatterji, N. S., Haque, S., and Hashimoto, T. (2022). Is importance weighting incompatible with interpolating classifiers? In *International Conference on Learning Representations*.

Wei, C., Lee, J. D., Liu, Q., and Ma, T. (2019). Regularization matters: Generalization and optimization of neural nets vs their induced kernel. In *Advances in Neural Information Processing Systems*.

Wen, Y., Jerfel, G., Muller, R., Dusenberry, M. W., Snoek, J., Lakshminarayanan, B., and Tran, D. (2021). Combining ensembles and data augmentation can harm your calibration. In *International Conference on Learning Representations*.

Wilson, A. G. and Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, D., Ye, Y., and Ruan, C. (2021). Understanding the role of importance weighting for deep learning. In *International Conference on Learning Representations*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

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

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. (2020). Cyclical stochastic gradient MCMC

for Bayesian deep learning. In *International Conference on Learning Representations*.

Supplementary materials: A Predictive View of Uncertainty Quantification in Deep Learning

A Formal statement and proof of Proposition 1

A.1 Both DE and stabilized BB converge to the max margin solution

Several works have considered the limiting margin behavior of homogeneous neural networks (Wei et al., 2019; Lyu and Li, 2020; Ji and Telgarsky, 2020; Xu et al., 2021). The margin for a single datapoint $z_i = (x_i, y_i)$ is defined as $\gamma_i = y_i f_\theta(x_i)$, and the margin for the entire dataset as $\gamma_{\min}(\theta) = \min_i \gamma_i$. In the case of L-homogeneous neural networks—i.e., networks where $f_{c\theta}(x) = c^L f_\theta(x)$ for some $L > 0$ and all $c > 0$—the margin $\gamma_{\min}(\theta)$ scales linearly with $||\theta||_2^L$, so we consider the normalized margin,

$$\tilde{\gamma}(\theta) = \gamma_{\min}\left(\frac{\theta}{||\theta||_2}\right) = \frac{\gamma_{\min}(\theta)}{||\theta||_2^L}$$

Under certain conditions, this normalized margin has been shown to converge to a max-margin solution γ^* (Wei et al., 2019; Lyu and Li, 2020; Ji and Telgarsky, 2020). Xu et al. (2021) show that such behavior can also be found when we incorporate weights in empirical loss minimization, where we have a loss of the form form $\mathcal{L}(\theta) = \sum_i w_i \ell_\theta(z_i) + \lambda||\theta||^r$ for some $r > 0$ and $\lambda \to 0$, referred to as weak regularization.

Below, we consider the binary classification setting, with $y_i \in \{-1, +1\}$. Extension to the multi-class setting is straightforward. Following Xu et al. (2021), we make the following assumptions:

Assumptions

A1 The loss takes the form $\ell_\theta(z_i) = \ell(y_i f_\theta(x_i))$, where $\ell(\cdot)$ has exponential tail behavior s.t. $\lim_{u\to\infty}\ell(-u) = \lim_{u\to\infty}\Delta\ell(-u) = 0$.

A2 $f_\theta(x_i)$ is L-homogeneous, i.e. $f_{c\theta}(x) = c^L f_\theta(x)$ for some $L > 0$, all $c > 0$, and all x and θ.

A3 $f_\cdot(x)$ is β-smooth and l-Lipschitz on $\mathbb{R}^d$ for all x.

A4 The data are separable by f_θ, and this condition can be reached via gradient descent. Further, $y_i f_{\theta^*}(x_i) \geq \gamma^* > 0$ for each i.

A5 The weights w_i are bounded, s.t. $w_i \in [1/M, M]$ for some positive M.

Lemma 1 (Proposition 3 of Xu et al. (2021))**.** *If Assumptions A1-A5 hold, then* $\lim_{\lambda\to 0}\tilde{\gamma}(\theta) \to \gamma^*$*, i.e.* f_θ *converges to the max margin solution.*

Cross-entropy loss meets assumption A1; see Xu et al. (2021) for more general sufficient descriptions of exponential tail behavior. Assumption A2 is met by feedforward and convolutional neural networks with ReLU activations and no bias terms. Assumption A4 is common in image datasets where we can achieve perfect train set classification. This is often the case in practice; for example, the MNIST and FashionMNIST training datasets used in this paper can be perfectly classified using a CNN with no bias terms.

DE trivially meets Assumption A5 since the weights are all $1/n$. In the Bayesian bootstrap setting, we ensure assumption A5 is met via the following definition of stabilized BB weights:

$$\begin{aligned}(\tilde{w}_1, \ldots, \tilde{w}_n) \sim &\text{Dirichlet}(1, \ldots, 1)\\ w_i = &\frac{\tilde{w} + \eta}{n + \eta},\end{aligned}$$

where $\eta = 1/(M - n)$ for some $M > n$.

It therefore follows from Lemma 1 that both DE and stabilized BB converge to the max margin classifier, provided assumptions A2, A3 and A4 are met. As noted by Xu et al. (2021), in practice the smoothness condition in A3 is not met when using ReLU activations; however, they find that in practice using ReLU rather than a smoother activation function makes little difference. Similarly, incorporating bias terms in a CNN with ReLU activations violates the homogeneity assumption A2; however in practice we found little difference when bias terms are included.

A.2 Equivalency of DE and stabilized BB

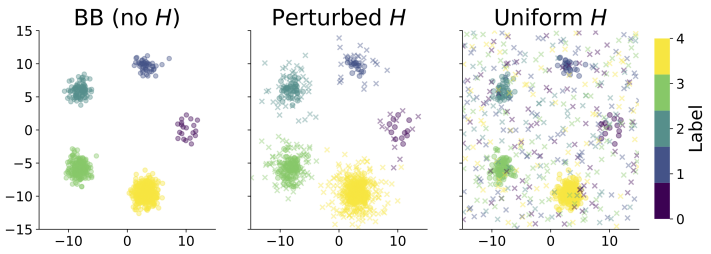
We restate Proposition 1 more formally:

Proposition 1. *Assume A1-5 are met, and inference is run until convergence of the normalized max margin. Then, any posterior sample obtained via stabilized BB could also have been obtained via DE, and vice versa.*

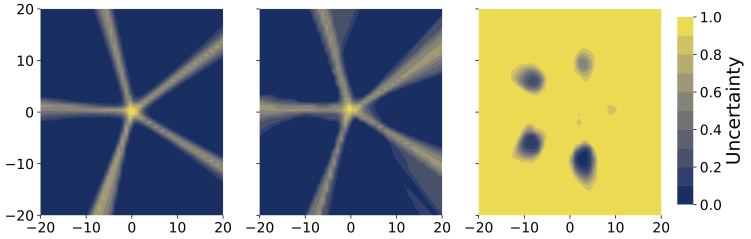
This is a direct consequence of Lemma 1: Any classifier that minimizes the randomly weighted loss used in stabilized BB will also minimize the unweighted loss.

Remark Proposition 1 implies that a sample from DE is a valid minimizer of stabilized BB. This is validated empirically in Section 6.1 and Appendix D: when we initialize a BB sample using a pretrained DE sample, we do not see any significant change in the resulting classifier.

However, Proposition 1 does not consider the probability of obtaining such a sample under the two schemes, when trained from a random initialization. Lemma 1 only considers asymptotic behavior of the margin on the training set; different weighting schemes will likely impact early-stage learning behavior which can impact *which* local minima the optimization ends up in. Indeed, Xu et al. (2021) show that weights can have an impact on out-of-sample performance in the finite-sample setting. In principle, this impact could lead to differing distributions of fitted parameters under DE and BB.



(a) $F_\infty^{(\mathrm{MMP})}$ sample from different MGs. Left: Bayesian bootstrap (BB); Middle: DP-MP with the base measure H being a perturbed distribution; Right: DP-MP with H being a uniform measure. Dots represent samples from observations and crosses are samples from the base measure. The label indexing starts from 0 to $K-1$.



(b) The corresponding predictive uncertainty landscape. BB and DP-MP with perturbed H both have uncertainty centered around decision boundary, with the later has greater uncertainty; DP-MP has high uncertainty in region where observations are absent or scarce.

Figure A.1: Illustration of Martingale posteriors. Different specifications of the future predictive distribution lead to different uncertainty quantification behaviors.

However, our experiments in Section 6.1 and Appendix D indicate that there is not a significant practical difference. When BB samples are initialized using the same random initialization as DE samples, we do see some variation in the resulting classifiers as a result of the different training dynamics; however the differences are small. This finding is in line with previous empirical work that shows importance weighting has no impact on the limiting behavior of neural networks, under exponentially-tailed losses (Byrd and Lipton, 2019; Wang et al., 2022).

We note that this result does not necessarily imply that the distribution over solutions will be the same under both paradigms: the weights before the loss terms will likely impact early-stage learning behavior which can impact *which* local minima the optimization ends up in. Indeed, Xu et al. (2021) show that weights can have an impact on out-of-sample performance in the finite-sample setting.

However, empirically we find there to be very little difference in the resulting behavior of neural networks trained under Bayesian bootstrapping and deep ensembles (Table 1/Table D.1 and Figure D.1). This finding is also supported by empirical work that shows importance weighting has no impact on the limiting behavior of neural networks, under exponentially-tailed losses (Byrd and Lipton, 2019; Wang et al., 2022).

B Illustrations of Martingale posteriors and synthetic experiment details

B.1 Illustration of martingale posteriors

We use a 2D synthetic experiment to illustrate how different specifications of future predictive distributions $\mathbb{P}_\infty$ can lead to different uncertainty quantification behaviors.

We randomly generate 5 clusters of 2D inputs of size $[20, 50, 100, 200, 500]$, each corresponding to a class $k \in [1, 2, 3, 4, 5]$. We consider three specifications of $\mathbb{P}_\infty$, all through specifying an exchangeable latent distribution F_∞ (see Section 2.1): (1) Bayesian Bootstrap (BB), (2) DP-MP with $c = 1$ and base measure $H =$ "Perturbed" – a data-driven model that injects random noise to observed inputs and preserves their labels, and (3) DP-MP

with $c = 1$ and $H =$ "Uniform" – an uninformative prior model that uniformly samples inputs and labels.

More specifically, in (2) the perturbed H generates a new pseudo sample $z' = (x', y')$ by choosing a random observation $z = (x, y)$ and then set $x' := x+\epsilon$ with $\epsilon \sim \mathcal{N}(0, 4)$ and $y' := y$; and in (3) the uniform H generates independent uniform pseudo samples $z' = (x', y')$ where $x' \sim \text{Uniform}[-15, 15]^2$, and $y' \sim \text{Categorical}(1, 2, 3, 4, 5)$.

We use a feed forward neural network with the hidden units size of $[16, 32, 16]$. We obtain their posterior samples $\{\theta^{(b)}\}_{b=1}^B$ according to Algorithm 1 with $B = 10$ and $T = n$, the total size of observations. In each simulation, the network is trained from a random initialization with a learning rate of 0.5 for 10,000 epochs, and no mini-batch is used.

The predictive uncertainty is computed as $1 - \sum_{k=1}^K p^2(y = k|z)$ (as suggested by Abe et al. (2022)) and is re-scaled to $[0, 1]$, where $p(\cdot|z)$ is evaluated by the mean of the B posterior predictive distributions.

Figure A.1a reflects a random sample of F_∞ from the three specifications. The corresponding landscape of predictive uncertainty is depicted in Figure A.1b. We can see that different specifications of future predictive distributions lead to different uncertainty quantification behaviors; for example, if we expect the future data to come from existing observations (as suggested by BB) or to be very similar to them (DP-MP with perturbed H), then there will be minimal uncertainty mostly concentrated around the decision boundary; however, if we expect almost "zero" prior knowledge (DP-MP with uniform H), then the uncertainty will be high except in the area with abundant evidence.

It is critical to emphasize that there is no singular, correct specification of a Martingale posterior unless we have complete knowledge of the true data generative process. The key takeaway from this synthetic example is to illustrate that one can incorporate prior beliefs by defining a future predictive distribution, which will lead to a more informed representation of uncertainty.

B.2 Experiment details for Figure 1 and results for varying α

In Figure 1, the synthetic data generation and the neural network architecture are the same as described in Appendix B.1. We obtain their posterior samples $\{\theta^{(b)}\}_{b=1}^B$ according to Algorithm 2 with $B = 10$ and $t_{mb} = n_{mb}$, and no standard data augmentation is used (i.e. $n_{mb} = n$). In each simulation, the network is trained from a random initialization with a learning rate of 0.5 for 10,000 epochs, and no mini-batch is used.

Here we also include the illustration of MixupMP with varying mixup parameter α in Figure B.1.

C Uncertainty calibration metrics

For uncertainty calibration metrics used in Section 6, we consider the expected calibration error (ECE, Naeini et al., 2015), over-confidence error (OE), and under-confidence error (UE).

ECE measures the difference between accuracy and confidence:

$$\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{n} |\text{Acc}(B_m) - \text{Conf}(B_m)| \tag{8}$$

where for $m = 1, \cdots, M$

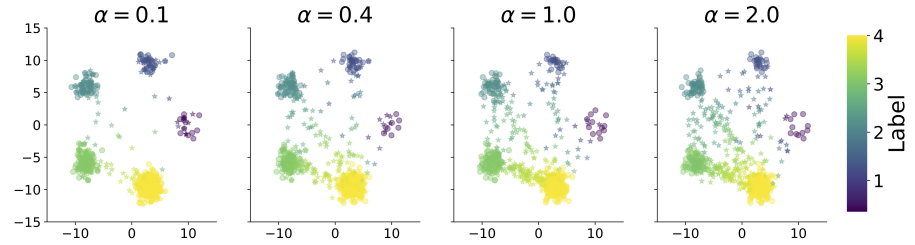
$$\text{Acc}(B_m) = \frac{1}{B_m} \sum_{x_i \in B_m} \mathbf{1}(\hat{y}_i = y_i), \quad \text{Conf}(B_m) = \frac{1}{B_m} \sum_{x_i \in B_m} \hat{p}_i, \tag{9}$$

each B_m is the m^{th} bin of samples whose prediction confidence falls into interval $(\frac{m-1}{M}, \frac{m}{M}]$, $\hat{y}_i$ is the prediction and $\hat{p}_i$ is the prediction conference. We use $M = 15$ throughout all experiments.

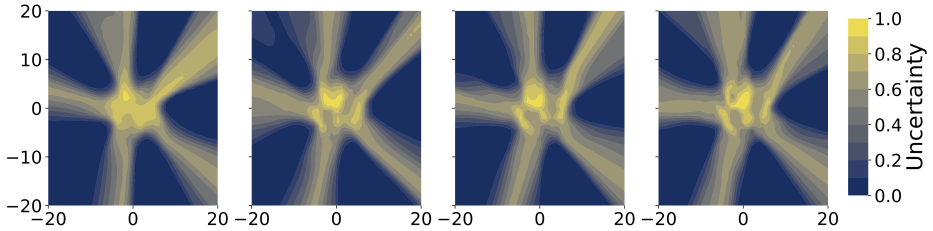
We further inspect the over-confidence and under-confidence quantification:

$$\text{OE} = \sum_{m=1}^M \frac{|B_m|}{n} \max\{\text{Conf}(B_m) - \text{Acc}(B_m), 0\} \tag{10}$$

$$\text{UE} = \sum_{m=1}^M \frac{|B_m|}{n} \max\{\text{Acc}(B_m) - \text{Conf}(B_m), 0\} \tag{11}$$



(a) $F_\infty^{(\mathrm{MMP})}$ sample from MixupMP with varying $\alpha \in [0.1, 0.4, 1.0, 2.0]$, $r = 1$ and $H_x^{(\mathrm{aug})} := \delta_x$. Dots represent samples from observations and crosses are samples from the base measure. The label space is extended to interval $[0, K-1]$.



(b) The corresponding predictive uncertainty landscape.

Figure B.1: Illustration of MixupMP on synthetic classification task ($K = 5$) with $r = 1.0$ and varying α. As α increases, $F_\infty^{(\mathrm{MMP})}$ samples are more concentrated on the middle of different data pairs, inducing wider predictive uncertainty bands around the decision boundary.

The study of OE and UE metrics are motivated by Thulasidasan et al. (2019), where they propose another version of over-confidence metric that weights each summand in Equation (10) by the confidence score.

D Equivalency of BB and DE: Details and additional results

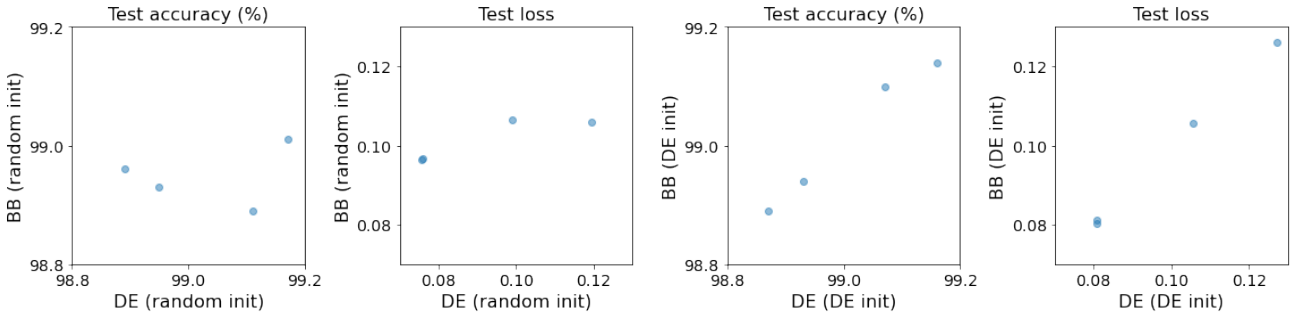
D.1 Experimental details

To compare BB and DE, we looked at the behavior of ensembles of neural networks trained on separable data. Specifically, we looked at the MNIST dataset (LeCun and Cortes, 2010, copyright the authors) and the Fashion MNIST (FMNIST) dataset (Xiao et al., 2017, MIT license). Each ensemble contains 4 independently trained neural networks. We used a convolutional neural network with two convolutional layers (with kernel size 5; 6 and 16 hidden channels; ReLU activation; and max pooling) and two fully connected layers (with 120 and 84 latent dimensions and ReLU activation).

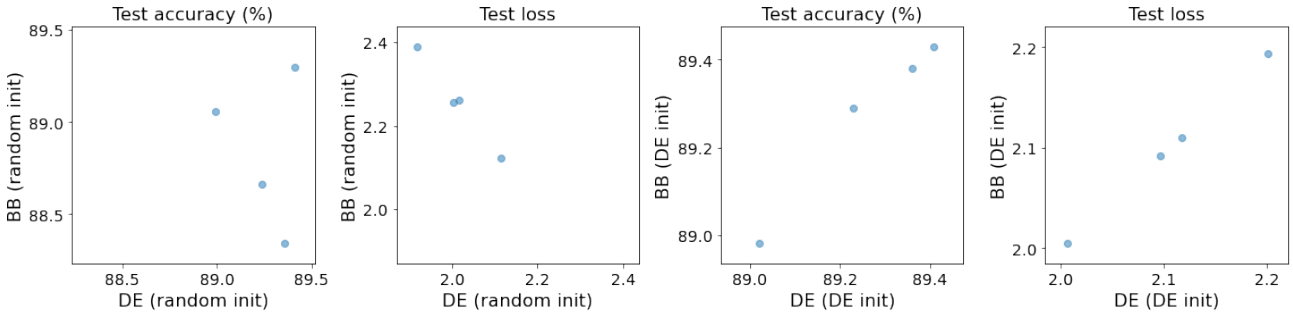
For the "random" initialization results, each ensemble member was initialized using one of four randomly generated initializations. Initializations were paired between the two methods (e.g., the initialization of the first DE ensemble member is the same of that of the first BB DE ensemble member). Models were trained using stochastic gradient descent with a learning rate of $5e^{-4}$, for 1000 epochs beyond obtaining 100% accuracy on the test set. These experiments were designed to explore the difference between BB and DE, given the same initialization.

The "DE" initialization results were obtained using ensembles where each member was initialized to the output of one of the DE "random" initialization ensemble members. These results were designed to assess Proposition 1, which claims that a solution for DE is also a solution for BB. If this were not the case, we would expect the BB solution to diverge from the DE solution. Since the initialization already achieves 100% training accuracy, models were trained using stochastic gradient descent with a learning rate of $5e^{-4}$, for 1000 epochs.

Table D.1 (an extended version of Table 1, with additional metrics included) shows the performance of the resulting ensembles. We see that there is a small difference between the BB and DE results with random initialization, which we hypothesize is due to early training conditions. As expected, we see almost no difference when initialized to a pre-trained DE solution.



(a) MNIST



(b) FashionMNIST

Figure D.1: Comparing individual ensemble members. Each point represents a pair of ensemble members from BB or DE with the same initialization (random or DE).

Figure D.1 shows the test set accuracies and losses of each ensemble member in the BB and DE ensembles, pair-matched to have the same initialization. As in Table D.1, when randomly initialized, we see that the models obtain similar solutions, but not exactly the same (left two plots of each row). However, when pre-initialized to a DE solution, we see little deviation between the DE and BB individual ensemble members (right two plots of each row).

Table D.1: Comparing Bayesian bootstrap (BB) with deep ensembles (DE). Models are either randomly initialized (same set of seeds for DE and BB), or initialized using a pretrained DE

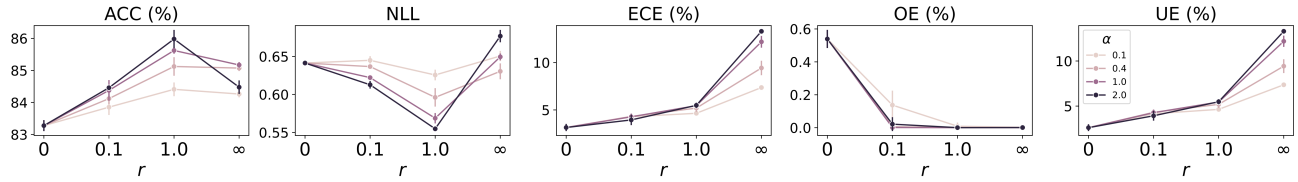
Dataset	Method	Init.	ACC (%)	ECE (%)	OE (%)	UE (%)	NLL
MNIST	DE	random	99.33	0.4092	0.2224	0.1868	0.035111
MNIST	BB	random	99.17	0.2357	0.2133	0.0224	0.037333
MNIST	DE	DE	99.33	0.3997	0.2185	0.1812	0.036600
MNIST	BB	DE	99.33	0.4079	0.2195	0.1884	0.036653
FMNIST	DE	random	91.52	2.3240	2.1323	0.1917	0.571902
FMNIST	BB	random	91.21	2.0132	1.8851	0.1281	0.620820
FMNIST	DE	DE	91.57	2.4219	2.1655	0.2564	0.592023
FMNIST	BB	DE	91.55	2.3866	2.1616	0.2249	0.591654

E Empirical study of MixupMP: experiment details and additional results

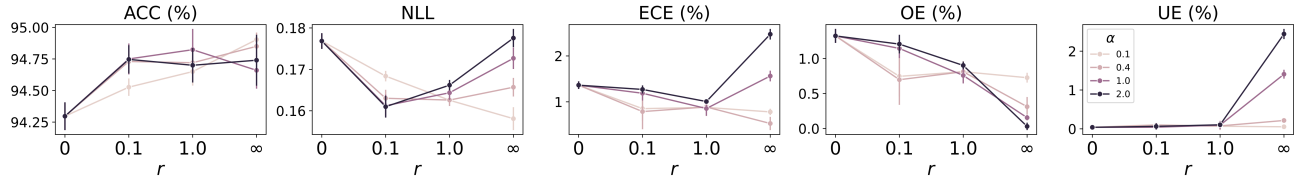
E.1 Implementation details.

We list implementation details of experiments in Sections 6.2 to 6.4.

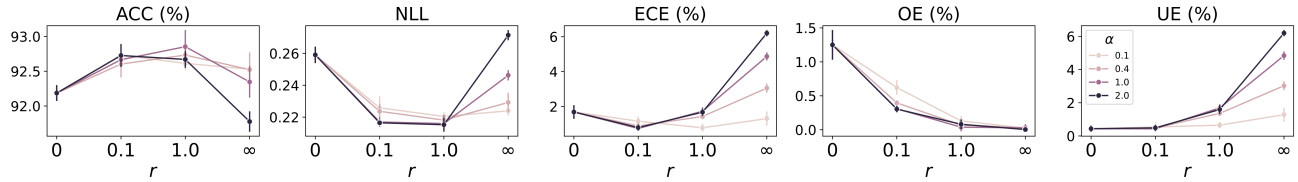
For CIFAR10 and CIFAR100 datasets (under MIT license), we use the Wide Resnet architecture with depth 28 and widen factor 10 for individual models (Zagoruyko and Komodakis, 2016), and the implementation from `https://github.com/meliketoy/wide-resnet.pytorch` which is also under an MIT license. We follow the



(a) Performance on CIFAR100 dataset



(b) Performance on FMNIST dataset (ResNet18)



(c) Performance on FMNIST dataset (CNN)

Figure E.1: MixupMP ablation study: Impact of α and r on test set performance on CIFAR100 (top row) and FMNIST (mid row: with ResNet18 and bottom row: with CNN). DE corresponds to MixupMP with $r = 0$; Mixup Ensemble corresponds to MixupMP with $r = \infty$. Solid lines are average values computed over 3 random runs, with error bars denoting 2 standard errors. The result for CIFAR10 is depicted in Figure 2.

training details from the original paper: "We use SGD with Nesterov momentum and cross-entropy loss. The initial learning rate is set to 0.1, weight decay to 0.0005, dampening to 0, momentum to 0.9 and minibatch size to 128. Learning rate dropped by 0.2 at 60, 120 and 160 epochs and we train for total 200 epochs."

For FMNIST dataset, we use the Resnet18 architecture (He et al., 2016), which is under an MIT license, and the implementation from `https://github.com/kefth/fashion-mnist/tree/master`. We additionally include the CNN architecture as an ablation (see results in Figure E.1c). Unless otherwise stated, the presented FMNIST results are obtained with Resnet18. The remaining training details are the same as the CIFAR experiment.

For implicit ensemble methods (namely MC Dropout, MixupMP-MC, Mixup-MC, CAMixup-MC), a dropout rate of 0.3 is used. All remaining methods do not use dropout.

CAMixup-MC requires using a validation set during training to evaluate the current ensemble's calibration behavior. We follow the practice from Wen et al. (2021) to separate 5% of data from the training set for validation purpose.

For Laplace method, we use the implementation from the official codebase `https://github.com/aleximmer/Laplace`.

For MixupMP and its variants, we set the pseudo sample batch size t_{mb} to the data batch size n_{mb} (described in Algorithm 2) throughout all experiments. Specifically, we used a batch size of 128.

Lastly, for all ensemble methods (either explicit or implicit ensemble), predictions are computed as the average of predictions of individual models.

E.2 Ablation study on hyperparameters and data augmentations

We include additional ablation results on varying ratio r and varying mixup parameter α on the other two datasets in Figure E.1. We observe similar trends to those found in the result from the CIFAR10 dataset, as is shown in Figure 2 and summarized in Section 6.2. However, the optimal choice of r and α with respect to

Table E.1: Full in-distribution test results on CIFAR10, CIFAR100 and FMNIST with standard errors included in the parenthesis. Bolded metrics are the best (within 2 standard errors) in each group of {single model, explicit ensemble, implicit ensemble}; $*$ indicates the best among all methods. The blue shaded rows correspond to results of our method MixupMP or its variants.

Dataset	CIFAR10			CIFAR100			FMNIST		
Metric	ACC (%)	NLL	ECE (%)	ACC (%)	NLL	ECE (%)	ACC (%)	NLL	ECE (%)
Single model									
NN	96.12(0.04)	0.1473(0.0012)	2.02(0.04)	80.82(0.08)	0.7816(0.0026)	4.78(0.08)	93.72(0.08)	0.2181(0.0024)	2.89(0.06)
MixupMP-single (r=0.1, α=0.1)	96.35(0.04)	0.1356(0.0010)	1.26(0.04)	81.30(0.06)	0.7712(0.0021)	3.66(0.11)	94.09(0.05)	0.1959(0.0010)	1.95(0.11)
MixupMP-single (r=0.1, α=0.4)	96.51(0.03)	0.1302(0.0009)	0.95(0.03)	81.85(0.08)	0.7535(0.0023)	3.54(0.06)	94.23(0.03)	0.1914(0.0012)	2.01(0.12)
MixupMP-single (r=0.1, α=1.0)	96.65(0.04)	0.1258(0.0012)	0.91(0.05)	82.22(0.06)	0.7322(0.0020)	3.23(0.08)	94.31(0.05)	0.1890(0.0011)	2.29(0.03)
MixupMP-single (r=0.1, α=2.0)	96.62(0.03)	0.1260(0.0008)	1.05(0.02)	82.47(0.06)	0.7201(0.0020)	3.24(0.11)	94.31(0.03)	0.1895(0.0009)	2.35(0.05)
MixupMP-single (r=1.0, α=0.1)	96.57(0.03)	0.1288(0.0010)	**0.46(0.04)**	82.09(0.05)	0.7352(0.0025)	2.72(0.09)	94.21(0.04)	0.1870(0.0010)	1.92(0.05)
MixupMP-single (r=1.0, α=0.4)	96.87(0.04)	0.1202(0.0011)	0.68(0.04)	82.86(0.07)	0.7046(0.0022)	1.66(0.08)	94.25(0.04)	0.1893(0.0010)	1.96(0.04)
MixupMP-single (r=1.0, α=1.0)	97.06(0.03)	0.1130(0.0008)	0.77(0.04)	83.27(0.05)	0.6769(0.0024)	**1.28(0.09)***	94.27(0.04)	0.1913(0.0011)	2.03(0.03)
MixupMP-single (r=1.0, α=2.0)	**97.27(0.03)**	**0.1055(0.0009)**	0.62(0.05)	**83.56(0.05)**	**0.6582(0.0012)**	**1.13(0.06)***	94.19(0.04)	0.1938(0.0009)	2.10(0.04)
Mixup (α=0.1)	96.40(0.03)	0.1354(0.0010)	0.82(0.05)	81.82(0.06)	0.7412(0.0029)	3.30(0.09)	**94.48(0.03)**	**0.1833(0.0010)**	1.79(0.03)
Mixup (α=0.4)	96.74(0.03)	0.1378(0.0011)	2.68(0.05)	82.69(0.05)	0.7156(0.0030)	5.11(0.20)	94.33(0.03)	0.1902(0.0008)	1.35(0.08)
Mixup (α=1.0)	96.88(0.04)	0.1566(0.0017)	5.36(0.18)	82.59(0.07)	0.7339(0.0021)	7.61(0.22)	94.20(0.04)	0.1933(0.0009)	**0.81(0.03)**
Mixup (α=2.0)	96.88(0.03)	0.1854(0.0016)	8.59(0.14)	81.83(0.09)	0.7614(0.0028)	8.54(0.19)	94.15(0.05)	0.1948(0.0008)	1.46(0.05)
Laplace	96.04(0.05)	0.1344(0.0010)	0.80(0.04)	80.95(0.18)	1.0107(0.0124)	22.96(1.02)	89.67(0.10)	0.4088(0.0045)	5.89(0.05)
Explicit ensemble (B=4)									
DE	96.83(0.02)	0.1090(0.0007)	0.78(0.04)	83.28(0.08)	0.6413(0.0003)	**3.14(0.15)**	94.30(0.05)	0.1768(0.0009)	1.36(0.03)
MixupMP (r=0.1, α=0.1)	97.01(0.05)	0.1042(0.0007)	**0.40(0.07)***	83.85(0.11)	0.6449(0.0023)	4.29(0.03)	94.53(0.03)	0.1684(0.0006)	0.85(0.13)
MixupMP (r=1.0, α=0.1)	97.17(0.02)	0.1014(0.0011)	1.13(0.07)	84.41(0.10)	0.6256(0.0030)	4.64(0.04)	94.65(0.05)	0.1625(0.0006)	0.88(0.09)
MixupMP (r=0.1, α=0.4)	97.04(0.03)	0.1009(0.0007)	0.55(0.03)	84.12(0.07)	0.6367(0.0008)	4.31(0.09)	94.73(0.03)	0.1630(0.0009)	**0.79(0.19)***
MixupMP (r=1.0, α=0.4)	97.39(0.02)	0.0951(0.0015)	1.67(0.03)	85.12(0.13)	0.5962(0.0056)	5.19(0.16)	94.72(0.02)	0.1626(0.0006)	0.89(0.03)
MixupMP (r=0.1, α=1.0)	97.13(0.04)	0.0966(0.0016)	0.59(0.02)	84.38(0.15)	0.6222(0.0007)	4.27(0.14)	94.75(0.06)	0.1612(0.0012)	1.19(0.07)
MixupMP (r=1.0, α=1.0)	97.57(0.05)	0.0918(0.0004)	1.78(0.09)	85.62(0.04)	0.5687(0.0030)	5.45(0.12)	**94.82(0.08)***	0.1644(0.0008)	0.85(0.07)
MixupMP (r=0.1, α=2.0)	97.13(0.03)	0.0969(0.0004)	**0.46(0.03)**	84.46(0.03)	0.6127(0.0021)	3.95(0.24)	94.75(0.05)	0.1610(0.0012)	1.27(0.04)
MixupMP (r=1.0, α=2.0)	**97.77(0.02)***	**0.0845(0.0010)***	1.67(0.08)	**85.98(0.13)***	**0.5548(0.0011)***	5.49(0.10)	94.70(0.07)	0.1662(0.0005)	1.01(0.00)
Mixup Ensemble (α=0.1)	97.03(0.02)	0.1099(0.0005)	1.91(0.05)	84.26(0.04)	0.6506(0.0029)	7.36(0.06)	**94.90(0.03)***	**0.1581(0.0013)***	0.78(0.03)
Mixup Ensemble (α=0.4)	97.32(0.09)	0.1178(0.0009)	3.86(0.02)	85.07(0.04)	0.6304(0.0048)	9.43(0.35)	**94.85(0.05)***	0.1657(0.0010)	**0.53(0.06)***
Mixup Ensemble (α=1.0)	97.47(0.01)	0.1395(0.0018)	6.55(0.12)	85.17(0.04)	0.6495(0.0018)	12.19(0.27)	94.66(0.07)	0.1727(0.0012)	1.56(0.05)
Mixup Ensemble (α=2.0)	97.55(0.02)	0.1717(0.0022)	9.71(0.18)	84.48(0.10)	0.6768(0.0036)	13.30(0.04)	**94.74(0.10)***	0.1776(0.0010)	2.47(0.05)
Implicit ensemble (B=20)									
MC Dropout	96.16(0.05)	0.1315(0.0011)	1.46(0.04)	80.83(0.25)	0.7451(0.0045)	3.69(0.07)	94.41(0.04)	0.1806(0.0004)	1.90(0.05)
MixupMP-MC (r=0.1, α=0.1)	96.49(0.10)	0.1204(0.0035)	0.72(0.11)	81.78(0.16)	0.7215(0.0012)	3.17(0.05)	94.57(0.06)	0.1711(0.0006)	1.42(0.06)
MixupMP-MC (r=1.0, α=0.1)	96.75(0.02)	0.1163(0.0014)	0.57(0.05)	82.08(0.22)	0.7102(0.0035)	2.76(0.33)	**94.76(0.08)***	0.1646(0.0005)	1.27(0.06)
MixupMP-MC (r=0.1, α=0.4)	96.62(0.04)	0.1180(0.0010)	**0.46(0.08)***	81.80(0.10)	0.7276(0.0019)	3.07(0.04)	**94.65(0.16)***	0.1674(0.0025)	1.44(0.10)
MixupMP-MC (r=1.0, α=0.4)	97.09(0.03)	0.1103(0.0011)	1.12(0.04)	83.32(0.11)	0.6825(0.0057)	4.92(0.13)	**94.81(0.05)***	**0.1619(0.0005)**	0.98(0.04)
MixupMP-MC (r=0.1, α=1.0)	96.77(0.01)	0.1132(0.0018)	**0.35(0.06)***	82.50(0.07)	0.7067(0.0033)	3.55(0.07)	**94.78(0.05)**	0.1648(0.0008)	1.38(0.06)
MixupMP-MC (r=1.0, α=1.0)	**97.15(0.08)**	0.1063(0.0013)	1.20(0.10)	**83.55(0.03)**	0.6520(0.0020)	4.23(0.26)	**94.86(0.05)***	**0.1624(0.0012)**	0.90(0.02)
MixupMP-MC (r=0.1, α=2.0)	96.58(0.05)	0.1145(0.0029)	0.42(0.02)	82.42(0.12)	0.7079(0.0045)	3.39(0.10)	94.72(0.07)	0.1651(0.0015)	1.46(0.02)
MixupMP-MC (r=1.0, α=2.0)	**97.27(0.04)**	**0.1005(0.0006)**	1.21(0.08)	**83.58(0.05)**	**0.6419(0.0043)**	4.21(0.10)	**94.81(0.06)***	**0.1619(0.0007)**	0.97(0.04)
Mixup-MC (α=0.1)	96.55(0.06)	0.1272(0.0013)	1.43(0.07)	81.77(0.20)	0.7263(0.0025)	3.46(0.38)	**94.84(0.05)***	0.1649(0.0007)	1.00(0.06)
Mixup-MC (α=0.4)	96.68(0.03)	0.1394(0.0013)	3.38(0.14)	82.52(0.06)	0.7235(0.0029)	7.49(0.21)	**94.86(0.06)***	0.1656(0.0008)	**0.55(0.05)***
Mixup-MC (α=1.0)	96.96(0.07)	0.1613(0.0012)	6.71(0.27)	82.26(0.05)	0.7331(0.0087)	8.98(0.76)	94.73(0.03)	0.1748(0.0007)	1.81(0.04)
Mixup-MC (α=2.0)	96.80(0.02)	0.2084(0.0045)	10.98(0.26)	81.75(0.08)	0.7616(0.0046)	10.71(0.20)	94.69(0.03)	0.1796(0.0007)	2.84(0.10)
CAMixup-MC (α=0.1)	95.82(0.05)	0.1529(0.0083)	**1.86(0.85)***	79.83(0.44)	0.7942(0.0250)	**2.44(0.40)**	94.13(0.06)	0.1808(0.0025)	1.13(0.04)
CAMixup-MC (α=0.4)	95.72(0.05)	0.1571(0.0080)	2.52(0.68)	80.49(0.14)	0.7647(0.0071)	**1.97(0.25)**	94.19(0.04)	0.1817(0.0019)	0.86(0.14)
CAMixup-MC (α=1.0)	95.94(0.03)	0.1466(0.0076)	1.63(0.62)	80.21(0.19)	0.7739(0.0122)	**2.05(0.20)**	94.31(0.05)	0.1788(0.0012)	0.97(0.09)
CAMixup-MC (α=2.0)	96.11(0.10)	0.1365(0.0070)	1.21(0.08)	80.19(0.11)	0.7780(0.0061)	2.14(0.04)	94.27(0.04)	0.1818(0.0020)	1.04(0.14)

a specific metric may vary across datasets. In practice one can leave out an additional validation dataset to determine those values.

E.3 Comparison to other methods

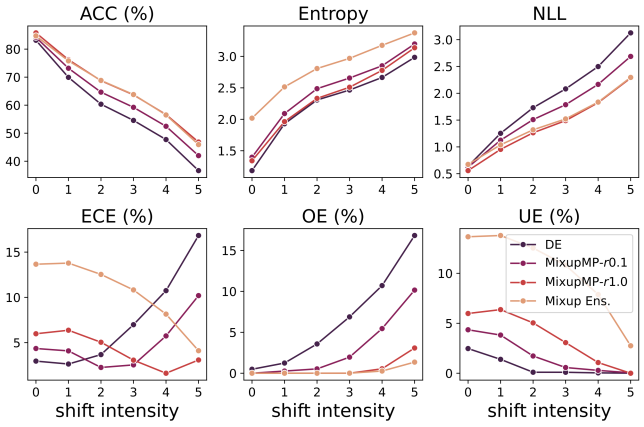
To complement the study in Section 6.3, we summarize the in-distribution test results for all methods with ablations on hyperparameters in Table E.1. In additional to the methods listed in Section 6.3, we include *MixupMP-single*, which is a single sample from MixupMP by running Algorithm 2 with $B = 1$.

We observe that MixupMP (including its variants MixupMP-single and MixupMP-MC) are the best or comparable to the best in terms of ACC and NLL in their corresponding group. The ECE performance of our method is either the best or close to best in each group, except that MixupMP-MC ($r = 0.4$) on FMNIST dataset and CAMixup-MC ($\alpha = 0.1, 0.4$ or 1.0) have better ECE. Among all methods, MixupMP via the explicit ensemble approach strikes the best balance in predictive performance and calibration.

E.4 Robustness to distribution shift

We conduct a comprehensive study of MixupMP in distribution shift / OOD (out-of-distribution) settings. For models trained on CIFAR10, we evaluate on CIFAR10-C (Hendrycks and Dietterich, 2018) which contain 19 corruption datasets, each with 5 intensity levels; For models trained on CIFAR100, we evaluate on CIFAR100-C (Hendrycks and Dietterich, 2018), which contain 21 corruption datasets, each with 5 intensity levels. Due to the

Figure E.2: Performance under distribution shift on CIFAR100. We plot various metrics against the distribution shift intensity ranging from 0 to 5, where 0 indicates no shift.



large size of the test set, we only report results using a single run of each model.

For MixupMP and its special cases DE and Mixup Ensemble, we investigate their performance under different shift levels. The results for CIFAR10-C are summarized in Figure 3, and the results for CIFAR100-C are summarized in Figure E.2. We observe similar trends in both plots; see detailed discussion in Section 6.4.

Full comparison. In Table E.2 we include a full comparison to all other methods listed in Section 6.3 and appendix E.3.

Across all methods, MixupMP with large values of r and α have superior OOD test performance in all metrics. Additionally, Mixup (single-model) and Mixup-MC also have favorable performance, especially in terms of ECE (they achieve the best ECE within their corresponding method group); however, Mixup Ensemble with a large α value (e.g. 1 or 2) can suffer from inflated ECE. Such results are expected, since with a large r, MixupMP (including Mixup Ensemble as a special case of $r = \infty$) has the prior belief that the future test data comes from a more uncertain domain away from the observations, which is the case in distribution shift settings.

While CAMixup-MC does well in in-distribution test set as shown in Table E.1, its OOD test performance is worse than MixupMP and Mixup Ensemble in most cases, especiallly in terms of ECE. This observation corroborates the findings in Wen et al. (2021). CAMixup adjusts the calibration behavior using only in-distributoin validation dataset, explaining its performance does not generalize well to OOD settings.

Lastly, we note that Laplace has inferior performance across all metrics, only slightly better than a single Neural Network (NN) and MixupMP-single with small r and α (in which case is a very close model to NN).

F Time and space complexity of methods described in the paper

On a per-epoch basis, the time and memory complexity of the MP methods described in this paper are of the same order of magnitude as the corresponding DE method. In the case of MixupMP, each minibatch is of length $t_{mb}+n_{mb}$, increasing the time and memory complexity by a constant factor. For BB, we have the same minibatch sizes as in DE; the additional cost of adding in weights is negligible.

Table E.2: Full out-of-distribution test results. Bolded values indicate the best result within the group in { single model, explicit ensemble, implicit ensemble}, and $*$ indicates the best among all methods. The blue shaded rows correspond to results of our method MixupMP or its variants.

Dataset	CIFAR10-C			CIFAR100-C		
Metric	ACC (%)	NLL	ECE (%)	ACC (%)	NLL	ECE (%)
Single model						
NN	75.50	1.1287	16.03	50.81	2.4562	15.93
MixupMP-single (r=0.1, α=0.1)	77.21	0.9422	12.91	53.31	2.2401	12.60
MixupMP-single (r=0.1, α=0.4)	78.78	0.8188	10.55	54.03	2.1875	10.96
MixupMP-single (r=0.1, α=1.0)	79.67	0.7787	9.95	55.41	2.0825	9.29
MixupMP-single (r=0.1, α=2.0)	80.38	0.7479	9.29	55.60	2.0860	9.94
MixupMP-single (r=1.0, α=0.1)	79.96	0.7725	9.33	54.88	2.0636	8.24
MixupMP-single (r=1.0, α=0.4)	81.42	0.7134	7.97	57.31	1.9141	6.57
MixupMP-single (r=1.0, α=1.0)	81.95	0.6887	7.48	58.49	1.8216	5.79
MixupMP-single (r=1.0, α=2.0)	**83.02**	0.6324	6.06	**58.94**	**1.7755**	5.67
Mixup (α=0.1)	80.77	0.7046	6.96	54.65	2.0589	7.72
Mixup (α=0.4)	81.66	0.6625	4.11	57.24	1.8749	4.78
Mixup (α=1.0)	82.04	**0.6286**	**2.34**	57.52	1.8255	**1.63**
Mixup (α=2.0)	80.75	0.6786	4.70	57.99	1.7932	2.16
Laplace	75.87	0.8933	11.98	51.26	2.3194	15.53
Explicit ensemble (B=4)						
DE	77.03	0.8968	10.29	53.84	2.1383	7.65
MixupMP (r=0.1, α=0.1)	78.79	0.7410	7.01	56.35	1.9735	4.78
MixupMP (r=1.0, α=0.1)	81.64	0.6174	3.99	57.82	1.8428	1.60
MixupMP (r=0.1, α=0.4)	80.53	0.6512	4.70	56.86	1.9375	3.94
MixupMP (r=1.0, α=0.4)	83.46	0.5558	1.82	60.39	1.7103	**1.12***
MixupMP (r=0.1, α=1.0)	81.24	0.6281	4.76	58.17	1.8669	2.72
MixupMP (r=1.0, α=1.0)	83.90	0.5371	1.61	61.82	1.6101	1.82
MixupMP (r=0.1, α=2.0)	82.02	0.6036	4.12	58.30	1.8550	3.28
MixupMP (r=1.0, α=2.0)	**85.12***	**0.4840***	**1.03***	**62.43***	**1.5632***	2.43
Mixup Ensemble (α=0.1)	82.80	0.5649	1.30	57.65	1.8422	1.29
Mixup Ensemble (α=0.4)	83.91	0.5468	3.27	60.56	1.6720	2.81
Mixup Ensemble (α=1.0)	84.05	0.5378	5.43	61.44	1.6322	7.17
Mixup Ensemble (α=2.0)	82.94	0.5886	9.30	62.18	1.6021	9.37
Implicit ensemble (B=20)						
MC Dropout	**82.87**	0.6778	8.72	50.58	2.4267	15.68
MixupMP-MC (r=0.1, α=0.1)	77.17	0.8627	10.66	52.73	2.2254	10.52
MixupMP-MC (r=1.0, α=0.1)	79.55	0.7213	7.32	54.27	2.1069	8.31
MixupMP-MC (r=0.1, α=0.4)	78.21	0.7938	8.94	53.42	2.1588	8.59
MixupMP-MC (r=1.0, α=0.4)	81.49	0.6429	4.94	56.24	1.9447	4.81
MixupMP-MC (r=0.1, α=1.0)	79.28	0.7276	7.47	54.30	2.1346	8.80
MixupMP-MC (r=1.0, α=1.0)	81.56	0.6472	4.60	57.40	1.8638	4.72
MixupMP-MC (r=0.1, α=2.0)	79.60	0.7324	7.58	54.71	2.0705	7.69
MixupMP-MC (r=1.0, α=2.0)	81.58	0.6619	5.40	**57.82**	**1.8104**	3.27
Mixup-MC (α=0.1)	79.50	0.7097	5.71	53.65	2.1018	7.61
Mixup-MC (α=0.4)	81.49	**0.6356**	**2.37**	55.75	1.9766	3.09
Mixup-MC (α=1.0)	80.86	0.6581	2.98	56.54	1.8723	**2.33**
Mixup-MC (α=2.0)	80.27	0.6964	6.19	56.47	1.8432	2.77
CAMixup-MC (α=0.1)	78.33	0.7711	6.22	52.86	2.1693	10.39
CAMixup-MC (α=0.4)	77.72	0.8392	7.44	53.07	2.1843	11.49
CAMixup-MC (α=1.0)	78.68	0.7890	7.11	52.28	2.2467	11.96
CAMixup-MC (α=2.0)	79.52	0.7728	8.49	52.68	2.2535	11.94

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] See Section 4
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] See Appendix F
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] `https://github.com/apple/ml-MixupMP`.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] See Appendix A.
   (b) Complete proofs of all theoretical results. [Yes] See Appendix A
   (c) Clear explanations of any assumptions. [Yes] See Appendix A.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] `https://github.com/apple/ml-MixupMP`.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] See relevant sections of the appendix
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] The metric definitions and the number of runs are described in the experiment section and appendix. The error bars in figures denote 2 standard errors.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Experiments were carried out using Apple internal clusters. We do not include full compute information for privacy reasons; however none of our results require a specific infrastructure.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] See Appendix D and Appendix E.
   (b) The license information of the assets, if applicable. [Yes] See Appendix D and Appendix E.
   (c) New assets either in the supplemental material or as a URL, if applicable. [No]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]