Greedy Bayesian Posterior Approximation with Deep Ensembles

Aleksei Tiulpin*
Department of Computer Science, Aalto University
Faculty of Medicine, University of Oulu
Finland

Matthew B. Blaschko
Centre for Processing Speech and Images
Department of Electrical Engineering
KU Leuven, Belgium

Abstract

Ensembles of independently trained neural networks are a state-of-the-art approach to estimate predictive uncertainty in Deep Learning, and can be interpreted as an approximation of the posterior distribution via a mixture of delta functions. The training of ensembles relies on non-convexity of the loss landscape and random initialization of their individual members, making the resulting posterior approximation uncontrolled. This paper proposes a novel and principled method to tackle this limitation, minimizing an f-divergence between the true posterior and a kernel density estimator in a function space. We analyze this objective from a combinatorial point of view, and show that it is submodular with respect to mixture components for any f. Subsequently, we consider the problem of ensemble construction, and from the marginal gain of the total objective, we derive a novel diversity term for training ensembles greedily. The performance of our approach is demonstrated on computer vision out-of-distribution detection benchmarks in a range of architectures trained on multiple datasets. The source code of our method is publicly available at `https://github.com/MIPT-Oulu/greedy_ensembles_training`.

1 Introduction

Estimation of predictive uncertainty is one of the most important challenges to solve in Deep Learning (DL). Applications in finance, medicine and self-driving cars are examples where reliable uncertainty estimation may help to avoid substantial financial losses, improve patient outcomes, or prevent fatal accidents [1]. However, to date, despite rapid progress, there is a lack of principled methods that reliably estimate the predictive uncertainty of deep neural networks (DNNs). As such, the most practical and empirically best-performing approaches is based on training a series of independent randomly initialized DNNs – *Deep Ensembles* (DE) [2, 3].

Recent studies, e.g. by Wilson and Izmailov [3] interpret ensembles as an approximation of predictive posterior. While this interpretation is correct from a Bayesian point of view, obtaining individual ensemble members via maximum a posteriori probability (MAP) estimation, may not lead to obtaining good coverage of the full support of the *function space* posterior distribution, and naturally has arbitrary bad theoretical approximation guarantees.

In this work, we propose *a novel and principled* methodology for approximate function space posterior inference in DNNs. Contrary to the mainstream BDL approach, which is based on defining a posterior distribution over the model parameters [4], we take a functional view, which allows us to treat the problem of training ensembles from combinatorial optimization prospective. This paper is a short version of [5].

*A part of this work was done at KU Leuven. Correspondence: `aleksei.tiulpin@aalto.fi`

Preprint. Under review.

2 Preliminaries

Consider an ensemble to be parameterized by a set of functions $Z = \{z_m\}_{m=1}^M \subset \mathcal{F}$, where $\mathcal{F}$ is a class of continuous functions, $z_m : \mathbb{R}^d \to \mathbb{R}^c$, with d the dimensionality of the input data, and c the dimensionality of the output. When training ensembles, we generally want to solve the following optimization problem:

$$\min_{Z,|Z|=M} \mathcal{R}(Z) - \Omega_{\lambda_M}(Z), \tag{1}$$

where $\mathcal{R}(Z) = \frac{1}{N}\sum_{i=1}^N \ell\left(\frac{1}{M}\sum_{m=1}^M z_m(x_i), y_i\right)$[2] is the empirical risk of the ensemble, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a loss function, $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ is a training dataset of size N, and $\Omega_{\lambda_M}(Z)$ is some diversity-promoting term, with diversity regularization strength λ_M. Let is now introduce the main notions of submodular analysis, a powerful tool that enables the analysis of the optimization of set functions.

Definition 1 (Submodularity)**.** *A function $f : 2^V \to \mathbb{R}$, for the power set of a base set V, is submodular if for all $A \subseteq B \subset V$ and $x \in V \setminus B$*

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B). \tag{2}$$

Definition 2 (Supermodularity and modularity)**.** *A set function is called supermodular if its negative is submodular, and modular if it is both submodular and supermodular.*

3 Submodular analysis of f-divergences

Main result We now consider the problem of approximating a Bayesian posterior via minimization of an f-divergence.

Consider some density $p(z)$ over continuous functions. We define $q_M(z) = \frac{1}{M}\sum_{m=1}^M K(d(z, z_j))$, $K_j(z) := K(z, z_j)$ is a kernel centered at z_j used to approximate the modes of $p(z)$.

Theorem 1. *Any f-divergence*

$$D_f(p||q_M) = \int f\left(\frac{p(z)}{\frac{1}{M}\sum_{j=1}^M K_j(z)}\right)\frac{1}{M}\sum_{m=1}^M K_m(z)dz \tag{3}$$

between a distribution $p(z)$ and a normalized mixture of M kernels with equal weights is supermodualar in a cardinality-fixed setting, assuming that $\forall z \max_{q_M} D_f(p(z)||q_M(z)) < \infty$.

Minimization of (3) is equivalent to a cardinality-constrained maximization of a non-monotone submodular function of $Z = \{z_1, \ldots, z_M\}$.

Marginal gains Although submodular optimization has natural parallel extensions and associated approximation guarantees, due to the simplicity of presentation, we focus in this paper on forward greedy selection, which requires the computation of the marginal gain on the objective function $F(Z)$, i.e. $\Delta(z_k|Z) = F(Z \cup \{z_k\}) - F(Z)$.

Proposition 1. *Consider $C = \max D_f(p||q_M)$, where $D_f(p||q_M)$ is an arbitrary f-divergence between some distribution $p(z)$ and a mixture of kernels $q_M(z) = \frac{1}{M}\sum_{j=1}^M K_j(z)$, and $D_f(p||q_M) < \infty$. Then, maximization of a marginal for $-D_f(p||q_M) + C$ at a step k of a greedy algorithm corresponds to*

$$\arg\max_{z_k} \Delta(z_k|Z) = \arg\min_{z_k} \mathbb{E}_{z\sim K_k(z)} f\left(\frac{p(z)}{\frac{1}{M}\sum_{j=1}^k K_j(z)}\right). \tag{4}$$

4 Objective function

We consider parametric functions $z_\theta : \mathbb{R}^d \to \mathbb{R}^c$. We also consider $f(x) = -\log x$ to be a generator for the f-divergence. $p(z_\theta|\mathcal{D}) \propto p(z_{\theta_1}, \ldots, z_{\theta_M}|\mathcal{D}) \propto \prod_{m=1}^M p(\mathcal{D}|z_{\theta_m})p(z_{\theta_m})$, where θ_m

[2] In practice we apply Jensen's inequality, and set $\mathcal{R}(Z) = \frac{1}{M}\sum_{m=1}^M \frac{1}{N}\sum_{i=1}^N \ell(z_m(x_i), y_i)$.

are parameters, $p(\mathcal{D}|z_\theta)$ the likelihood and $p(z_\theta)$ the prior; $p(\mathcal{D}|z_\theta) \propto \prod_{i=1}^{n} \exp(-\ell(z_\theta(x_i), y_i))$, and $p(z_\theta) \propto \exp(-\lambda\|\theta\|_2^2)$. To solve the problem defined by (4) in the context of Bayesian posterior approximation, we define the kernel density components via generalized exponential kernels $K_j(z_\theta) \propto \exp(-\lambda_M d(z_\theta, z_{\theta_j})^2)$, where λ_M is proportional to the kernel width. Furthermore, instead of full integration in (4), we consider a MAP-like point estimate. Therefore, at a k^{th} greedy step, we minimize

$$J(\theta_k) = \underbrace{\mathbb{E}_{(x,y)\sim p(x,y)}\ell(z_{\theta_k}(x), y) + \lambda\|\theta_k\|_2^2}_{\text{Marginal gain on } \mathcal{R}(Z)} + \underbrace{\log \sum_{j=1}^{k-1} \exp\left(-\frac{\lambda_M}{M} d(z_{\theta_k}, z_{\theta_j})^2\right)}_{\text{Marginal gain on } \Omega_{\lambda_M}(Z)} \tag{5}$$

to obtain a new ensemble member.

Based on hardness results for function norm computation [6], computing the marginal gain on the diversity term in (5) is non-trivial. Here, we use the following sampling-based approximation.

$$\log \sum_{j=1}^{k-1} \exp\left(-\frac{\lambda_M}{M} \mathbb{E}_{x\sim p^*(x)} \|z_{\theta_k}(x) - z_{\theta_j}(x)\|_2^2\right), \tag{6}$$

where $p^*(x)$ is a weighting distribution. In our experiments, we fit a pixel-wise Gaussian to the data, and set $p^*(x)$ to have $\times 5$ larger standard deviation (Appendix A).

5 Experiments

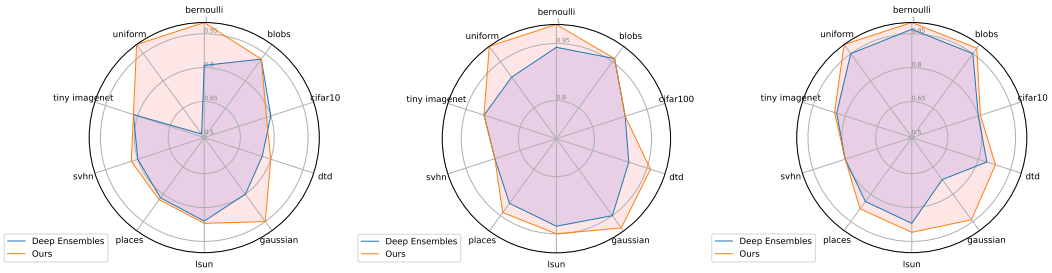
Datasets and models. We ran our main experiments on CIFAR10, CIFAR100 [7] and SVHN [8] in-distribution datasets. Our OOD detection benchmark included CIFAR10, CIFAR100, DTD [9], SVHN [8], LSUN [10], TinyImageNet [11], Places 365 [12], Bernoulli noise images, Gaussian noise, random blobs image, and uniform noise images. The composition of the benchmark was inspired by the work of Hendrycks et al. [13]. We excluded the in-distribution datasets for each of the settings, resulting in a total of 10 OOD datasets for each in-distribution dataset. The full description of the benchmark is shown in Appendix A.2. The experiments were conducted using ResNet164 (pre-activated version; denoted as PreResNet164) [14], VGG16 (with batch normalization [15]; denoted as VGG16BN) [16], and WideResNet28x10 [17]. Other relevant details are shown in Appendix B.1.

Model selection and metrics. We used mutual information (MI) between the distribution of the predicted label $\hat{y}$ for the point $\hat{\mathbf{x}}$ and the posterior distribution over functions $p(f|\mathcal{D})$, to evaluate the *epistemic uncertainty* (Appendix A.3) and reported the area under the ROC curve (AUC) and area under the precision-recall (PR) curve, i.e. average precision (AP) to quantify the OOD detection performance. Furthermore, we computed the false positive rate at 95% true positive rate (FPR95).

Out of distribution detection results We present aggregated results for all the models and in-distribution datasets in Table 1. It is clear that on average (across OOD datasets), our method is substantially better than DE. This holds for all the architectures and in-distribution datasets. We show the expanded version of all the OOD detection results in Appendix B.3. An example of these results is shown in Figure 1 for all the models trained on CIFAR 100. Here, one can see that our method is at least similar to DE, and substantially better overall.

6 Conclusion

In this paper, we have introduced a novel paradigm for Bayesian posterior approximation in Deep Learning using greedy ensemble construction via submodular optimization. We have proven a new general theoretical result, which shows that minimization of an f-divergence between some distribution and a kernel density estimator has approximation guarantees, and can be done greedily. We then derived a novel coverage promoting diversity term for ensemble construction. The results presented in this paper, demonstrate that our method outperforms DE [2], on a range of benchmarks. An extended version of this workshop paper can be found in [5]. Our code is available at `https://github.com/MIPT-Oulu/greedy_ensembles_training`.



(a) PreResNet164 (b) VGG16BN (c) WideResNet28x10

Figure 1: Out-of distribution detection results on CIFAR 100 for 3 different architectures (read column-wise). Here, we show AUC values from 0.5 to 1 averaged across 5 seeds.

Table 1: Averaged metrics across 10 OOD datasets.

Model	Dataset	Deep Ensembles			Ours		
		AUC (↑)	AP (↑)	FPR95 (↓)	AUC (↑)	AP (↑)	FPR95 (↓)
PreResNet164	C10	0.94	0.92	0.17	**0.95**	**0.95**	**0.14**
	C100	0.79	0.80	0.47	**0.88**	**0.88**	**0.40**
	SVHN	0.99	0.97	0.02	**1.00**	**0.98**	**0.01**
WideResNet28x10	C10	0.95	0.94	0.15	**0.96**	**0.96**	**0.12**
	C100	0.86	0.85	0.36	**0.90**	**0.91**	**0.30**
	SVHN	0.99	0.96	0.03	**1.00**	**0.99**	**0.01**
VGG16BN	C10	0.92	0.91	0.23	**0.95**	**0.95**	**0.18**
	C100	0.83	0.82	0.45	**0.89**	**0.90**	**0.36**
	SVHN	0.99	0.96	0.02	**1.00**	**0.98**	0.02

References

[1] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[2] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[3] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[4] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

[5] Aleksei Tiulpin and Matthew B Blaschko. Greedy Bayesian posterior approximation with deep ensembles. *arXiv preprint arXiv:2105.14275*, 2021.

[6] Amal Rannen-Triki, Maxim Berman, Vladimir Kolmogorov, and Matthew B Blaschko. Function norms for neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[7] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[8] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[9] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[10] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[11] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. CS 231N, Stanford University, 2015.

[12] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[13] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

[17] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[18] Prem Melville and Raymond J Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[19] 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*, pages 248–255. Ieee, 2009.

[20] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. NIPS'18, page 7047–7058, Red Hook, NY, USA, 2018. Curran Associates Inc.

[21] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, 2018.

[22] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, volume 2, 2019.

[23] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

A Implementation details

A.1 Practical implementation of the algorithm

Weighting distribution for the diversity term We propose the following simple heuristic, defining $p^*(x)$ as a normal distribution $\mathcal{N}(\mu_\mathcal{D}, \alpha \cdot \Sigma_\mathcal{D})$ of dimensionality, corresponding to the training data. The covariance $\Sigma_\mathcal{D}$ for this distribution is set to be diagonal, such that the variance for every dimension j is $\Sigma_\mathcal{D}[j,j] = (\alpha \cdot \sigma_j)^2$, where $\alpha > 1$ is a scaling parameter, and σ_j^2 is a variance of the dimension j computed from samples of the training dataset $\mathcal{D}$. Similarly, $\mu_\mathcal{D}$, the vector of expected values for every dimension, is also computed from the training data. Finally, the hyperparameter $\alpha = 5$ was found to work well, and we thus report all the experimental results with it fixed. We note that a similar technique, but for *in-distribution* data generation has been used earlier in [18].

The resulting algorithm The resulting, computationally tractable optimization algorithm for ensembles, which minimizes marginal gains (4), is shown in Algorithm 1. For simplicity, we omit the snapshot selection step, i.e. early stopping.

We note that contrary to the general Random Greedy method, shown in the main text, we can resort to a method with complexity of $\mathcal{O}(k)$. This is achieved through the fact that a uniform selection of the elements maximizing the marginal gain can be avoided, since at each greedy step, we initialize the new models randomly before maximizing the marginal gain. Another performance improvement can be gained by storing the evaluations $z_j(x_i)\ \forall j = 1, \ldots, k-1$ in memory before executing each k^{th} step.

We report here also one important practical trick, which we found important during the training. Specifically, freezing the batch normalization layers [15] before computing the diversity term turned out to help the convergence substantially. We anticipate that the diversity term weighting distribution approximated as a simple multivariate Gaussian with diagonal covariance may be corrupting the batch norm statistics. We thus think that using other, more sophisticated techniques for generating the weighting distribution samples might provide better results.

Algorithm 1 $\mathcal{O}(k)$ Random Greedy algorithm for training ensembles of neural networks.

Input: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ – Dataset
Input: M – Size of the ensemble
Input: N – Number of iterations
Input: α – Variance parameter for $p^*(x)$
Input: N_b – Mini-batch size
$Z \leftarrow \emptyset$
$\mathcal{D}^* \leftarrow \{(x^*_, y^*_i)\}_{i=1}^{n} \sim \mathcal{N}(\mu_{\mathcal{D}}, \alpha \cdot \Sigma_{\mathcal{D}})$
while $|Z| < M$ **do**
  $k \leftarrow |Z|$;
  Randomly initialize z_{θ_k};
  for $i = 1$ **to** N **do**
    $\mathcal{D}_i \leftarrow \{(x_b, y_b)\}_{b=1}^{N_b} \sim \mathcal{D}$;
    $\mathcal{D}_i^* \leftarrow \{x_b^*\}_{b=1}^{N_b} \sim \mathcal{D}^*$;
    $L \leftarrow \mathbb{E}_{(\hat{x},\hat{y})\sim\mathcal{D}_i} \ell(z_{\theta_k}(\hat{x}), \hat{y}) + \lambda\|\theta_k\|_2^2$
    $\Omega \leftarrow 0$
    if $k > 1$ **then**
      for $j = 1$ **to** $k-1$ **do**
        $d_{i_j} \leftarrow \mathbb{E}_{x^*\sim\mathcal{D}_i^*} \|z_{\theta_k}(x^*) - z_{\theta_j}(x^*)\|_2^2$
      end for
      $\Omega \leftarrow \log \sum_{m=1}^{k-1} \exp(-\frac{\lambda_M}{M} d_{i_m})$
    end if
    Update θ using $\nabla_\theta\,(L + \Omega)$
  end for
  $Z \leftarrow Z \cup \{z_{\theta_k}\}$
end while
return Z

A.2 OOD detection benchmark

The OOD benchmark included 10 different datasets. Here, we used DTD [9], Gaussian noise, Bernoulli noise, and uniform noise datasets In addition, we used Places 365 [12], Tiny ImageNet [11, 19], and LSUN [10] datasets in the benchmark. For CIFAR10 as in-domain data, we added CIFAR100 and SVHN [8] to the benchmark. For CIFAR100 – CIFAR10 and SVHN. Finally, for SVHN, we added CIFAR10 and CIFAR100 as OOD datasets, making a total of 10 OOD datasets per 1 in-distribution dataset. The details about each of the datasets are shown in Table 2.

Before feeding the images to the network, we applied re-scaling of the intensity range by subtracting the in-domain dataset mean, and dividing by the in-domain dataset std.

Table 2: Description of the datasets used in all the exp. R indicates real images, S – synthetic.

Dataset	Type	# samples	Comment
Uniform	S	$25,000$	N/A
Gaussian		$25,000$	Generated once, used in all experiments
Blobs		$25,000$	N/A
Bernoulli		$25,000$	N/A
CIFAR10	R	$10,000$	Test set (not used in training)
CIFAR100		$10,000$	Test set (not used in training)
SVHN		$73,257$	Test set (not used in training)
Places 365		$10,000$	First $10,000$ images from the test set (sorted alphabetically)
TinyImageNet		$10,000$	Original validation set images
DTD		$5,640$	Release 1.0.1
LSUN		$10,000$	Test set

A.3 Epistemic uncertainty computation

We used the *epistemic uncertainty*, i.e. mutual information (MI) between the distribution of predicted label $\hat{y}$ for the point $\hat{\mathbf{x}}$ and the posterior distribution over functions $p(f|\mathcal{D})$, to evaluate the uncertainty [20, 21]. As a distribution over weights induces a distribution over functions, we approximate the MI as:

$$\mathcal{I}(\hat{y}; f|\hat{\mathbf{x}}, \mathcal{D}) = \mathcal{H}\left[\mathbb{E}_{p(\theta|\hat{\mathbf{x}},\mathcal{D})} p(\hat{y}|\theta, \hat{\mathbf{x}}, \mathcal{D})\right] - \mathbb{E}_{p(\theta|\hat{\mathbf{x}},\mathcal{D})} \mathcal{H}\left[p(\hat{y}|\theta, \hat{\mathbf{x}}, \mathcal{D})\right], \tag{7}$$

where $\mathcal{H}[\cdot]$ denotes the entropy. One can see that this metric can be efficiently computed from the predictions of an ensemble.

B Experiments

B.1 Experimental details

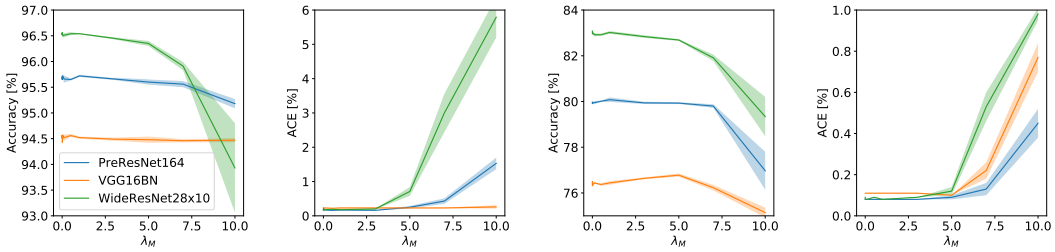
Model selection Contrary to the commonly used practice, we did not use CIFAR10/100 and SVHN test set sets for model selection. Neither did we use any OOD data. Instead, we used validation set accuracy (10% of the training data; randomly chosen stratified split) to select the models when optimizing the marginal gain. The best snapshot found using the validation data, was then selected for final testing. When selecting the models for evaluation on OOD data, we first evaluated ensembles on the in-distribution test set (Appendix B.2). Subsequently, we selected the highest λ_M that did not harm the test set (in-domain) performance (no overlap of confidence intervals defined as mean $\pm$ standard error). To provide additional information, we also analyzed adaptive calibration error (ACE) with 30 bins [22].

Hyper-parameters The main training hyper-parameters were adapted from [4] (see Table 3), but with additional modifications inspired by [20, 23], which helped to train the CIFAR models to state-of-the-art performance in only 100 epochs. As such, we first employed a warm-up of the learning rate (LR) from a value 10 times lower than the initial LR (LR_{init} in Table 3) for 5 epochs. Subsequently, after 50% of the training budget, we linearly annealed the LR to the value of $LR \times lr_{scale}$ until 90% of the training budget is reached, after which we kept the value of LR constant.

All models were trained using stochastic gradient descent with momentum of 0.9 and a total batch size of 128. We employed standard training augmentations – horizontal flipping, reflective padding to 34×34, and random crop to 34×34 pixels.

Model	$\mathbf{LR_{init}}$	Nesterov	Weight Decay	lr_{scale}
PreResNet164	0.1	Yes	0.0001	0.01
VGG16BN	0.05	No	0.0005	0.01
WideResNet28x10	0.1	No	0.0005	0.001

Table 3: Main hyper-parameters of all the used models.



(a) (b) (c) (d)

Figure 2: Relationship between accuracy, ACE, and λ_M ($M = 11$). Subplots (a) and (b) show the results for CIFAR10. Subplots (c) and (d) show the results for CIFAR100.

B.2 In-domain performance

In-distribution performance vs. diversity. Figure 2 provides an illustration of how the test set performance changes with λ_M on CIFAR data. One can see a general trend that when λ_M approaches M, the models lose the ability to make accurate predictions, which results in lower accuracy and poorer calibration. Interestingly, performance on the VGG model degrades much slower with λ_M compared to other architectures. Similar findings were also obtained for the SVHN dataset. Based on the test performance, we selected the models for further evaluation on OOD benchmark.

Best models' performance Table 4 shows the results of all the trained models on the in-domain data. One can see that the results between Deep Ensembles (DE) [2] do not differ significantly. We trained all these models according to the earlier specified hyper-parameters and the learning rate schedule. Models selected in Table 4 are used to report the results in the main experiments.

Table 4: In-domain performance on the test sets of CIFAR10/100 and SVHN for all the models used in the experiments ($M = 11$). We report mean and standard error over 5 random seeds for each of the models. Standard errors are reported if they are more than 0.01 across runs. DE indicates Deep Ensembles.

Architecture	Dataset	Method	Accuracy (%)	NLL $\times 100$	ACE (%)
PreResNet164	C10	DE	$95.70_{\pm 0.02}$	$13.28_{\pm 0.06}$	0.18
		Ours ($\lambda_M = 3$)	$95.66_{\pm 0.02}$	$13.18_{\pm 0.09}$	0.16
	C100	DE	$79.97_{\pm 0.04}$	$73.44_{\pm 0.17}$	0.08
		Ours ($\lambda_M = 5$)	$79.93_{\pm 0.04}$	$74.16_{\pm 0.91}$	$0.09_{\pm 0.01}$
	SVHN	DE	$99.46_{\pm 0.01}$	$2.34_{\pm 0.04}$	0.25
		Ours ($\lambda_M = 1$)	$99.38_{\pm 0.01}$	$3.16_{\pm 0.13}$	$0.81_{\pm 0.12}$
VGG16BN	C10	DE	$94.55_{\pm 0.02}$	$17.59_{\pm 0.05}$	0.23
		Ours ($\lambda_M = 5$)	$94.48_{\pm 0.06}$	$17.84_{\pm 0.15}$	$0.23_{\pm 0.01}$
	C100	DE	$76.32_{\pm 0.09}$	$91.07_{\pm 0.35}$	0.11
		Ours ($\lambda_M = 5$)	$76.78_{\pm 0.07}$	$89.10_{\pm 0.33}$	0.10
	SVHN	DE	$99.40_{\pm 0.01}$	$2.71_{\pm 0.02}$	$0.18_{\pm 0.01}$
		Ours ($\lambda_M = 1$)	$99.25_{\pm 0.01}$	$3.94_{\pm 0.10}$	$0.95_{\pm 0.12}$
WideResNet28x10	C10	DE	$96.56_{\pm 0.02}$	$10.76_{\pm 0.06}$	$0.16_{\pm 0.01}$
		Ours ($\lambda_M = 1$)	$96.54_{\pm 0.01}$	$10.99_{\pm 0.03}$	$0.18_{\pm 0.01}$
	C100	DE	$83.08_{\pm 0.09}$	$62.05_{\pm 0.18}$	0.09
		Ours ($\lambda_M = 1$)	$83.02_{\pm 0.06}$	$62.20_{\pm 0.13}$	0.08
	SVHN	DE	$99.45_{\pm 0.01}$	$2.53_{\pm 0.03}$	$0.43_{\pm 0.01}$
		Ours ($\lambda_M = 1$)	99.38	$2.87_{\pm 0.04}$	$0.46_{\pm 0.01}$

B.3 Detailed results

Detalized versions of the results presented in the main text are shown in Table 5. The corresponding λ_M coefficients are the same as in Table 4.

Table 5: CIFAR10 results. We report mean and standard error over 5 random seeds for each of the models. Standard errors are reported if they are more than 0.01 across runs. DE indicates Deep Ensembles.

Architecture	**OOD dataset**	**DE**			**Ours**		
		AUC (↑)	**AP (↑)**	**FPR95 (↓)**	**AUC (↑)**	**AP (↑)**	**FPR95 (↓)**
PreResNet164	bernoulli	$0.98_{\pm 0.01}$	$0.97_{\pm 0.01}$	$0.04_{\pm 0.01}$	**1.00**	**1.00**	**0.00**
	blobs	0.96	0.98	$0.12_{\pm 0.01}$	0.96	0.98	$0.12_{\pm 0.01}$
	cifar100	0.90	0.87	0.30	0.90	**0.88**	0.30
	dtd	0.93	0.83	$0.19_{\pm 0.01}$	**0.96**	**0.93**	**0.14**
	gaussian	$0.93_{\pm 0.01}$	$0.94_{\pm 0.01}$	$0.16_{\pm 0.01}$	$0.96_{\pm 0.02}$	$0.97_{\pm 0.02}$	$\mathbf{0.11_{\pm 0.03}}$
	lsun	0.93	0.89	0.20	**0.95**	**0.94**	**0.18**
	places	0.92	0.89	0.21	**0.94**	**0.93**	**0.19**
	svhn	0.94	0.99	0.16	**0.95**	0.99	**0.14**
	tiny imagenet	0.91	0.88	0.28	**0.92**	**0.89**	**0.26**
	uniform	$0.98_{\pm 0.01}$	$0.97_{\pm 0.01}$	$0.04_{\pm 0.01}$	**1.00**	**1.00**	**0.00**
VGG16BN	bernoulli	$0.94_{\pm 0.01}$	$0.95_{\pm 0.01}$	$0.11_{\pm 0.02}$	**1.00**	**1.00**	**0.00**
	blobs	0.96	0.98	0.16	0.96	0.98	**0.15**
	cifar100	0.89	0.86	0.34	0.89	0.86	0.34
	dtd	0.90	$0.77_{\pm 0.01}$	$0.25_{\pm 0.01}$	**0.96**	**0.92**	**0.18**
	gaussian	0.95	0.97	$0.14_{\pm 0.01}$	**0.99**	**0.99**	$\mathbf{0.05_{\pm 0.01}}$
	lsun	0.93	0.91	0.24	**0.95**	**0.94**	**0.21**
	places	0.91	0.90	$0.27_{\pm 0.01}$	**0.94**	**0.93**	**0.23**
	svhn	0.87	0.97	$0.28_{\pm 0.01}$	0.87	0.97	$0.27_{\pm 0.01}$
	tiny imagenet	0.90	0.88	0.33	0.90	0.88	**0.32**
	uniform	$0.90_{\pm 0.02}$	$0.89_{\pm 0.01}$	$0.16_{\pm 0.02}$	**1.00**	**1.00**	**0.00**
WideResNet28x10	bernoulli	1.00	1.00	0.00	1.00	1.00	0.00
	blobs	0.96	0.97	$0.11_{\pm 0.01}$	**0.97**	**0.98**	$0.10_{\pm 0.01}$
	cifar100	**0.92**	0.89	0.27	0.91	0.89	0.27
	dtd	0.93	$0.86_{\pm 0.01}$	$0.22_{\pm 0.01}$	**0.97**	**0.94**	$\mathbf{0.14_{\pm 0.01}}$
	gaussian	$0.96_{\pm 0.01}$	$0.97_{\pm 0.01}$	$0.09_{\pm 0.01}$	**1.00**	**1.00**	**0.00**
	lsun	0.93	0.91	0.20	**0.95**	**0.95**	$\mathbf{0.17_{\pm 0.01}}$
	places	0.93	0.91	0.21	**0.95**	**0.94**	**0.18**
	svhn	**0.96**	0.99	**0.12**	0.95	0.99	$0.13_{\pm 0.01}$
	tiny imagenet	0.92	0.90	0.27	**0.93**	**0.91**	**0.25**
	uniform	1.00	1.00	0.00	1.00	1.00	0.00

Table 6: CIFAR100 results. We report mean and standard error over 5 random seeds for each of the models. Standard errors are reported if they are more than 0.01 across runs. DE indicates Deep Ensembles.

Architecture	OOD dataset	DE			Ours		
		AUC (↑)	**AP (↑)**	**FPR95 (↓)**	**AUC (↑)**	**AP (↑)**	**FPR95 (↓)**
PreResNet164	bernoulli	$0.81_{\pm 0.03}$	$0.82_{\pm 0.03}$	$0.24_{\pm 0.04}$	$\mathbf{1.00}$	$\mathbf{1.00}$	$\mathbf{0.00}$
	blobs	$0.92_{\pm 0.01}$	0.95	$0.25_{\pm 0.02}$	$0.92_{\pm 0.02}$	$0.95_{\pm 0.01}$	$0.23_{\pm 0.03}$
	cifar10	$\mathbf{0.80}$	$\mathbf{0.76}$	$\mathbf{0.57}$	$0.78_{\pm 0.01}$	0.75	$0.67_{\pm 0.02}$
	dtd	$0.76_{\pm 0.01}$	$0.61_{\pm 0.01}$	$\mathbf{0.64_{\pm 0.01}}$	$\mathbf{0.80_{\pm 0.01}}$	$\mathbf{0.75_{\pm 0.01}}$	$0.78_{\pm 0.05}$
	gaussian	$0.80_{\pm 0.01}$	$0.83_{\pm 0.01}$	$0.37_{\pm 0.02}$	$\mathbf{0.95_{\pm 0.02}}$	$\mathbf{0.97_{\pm 0.01}}$	$\mathbf{0.16_{\pm 0.05}}$
	lsun	0.86	0.81	$\mathbf{0.45}$	$\mathbf{0.87}$	$\mathbf{0.85}$	$0.47_{\pm 0.01}$
	places	0.82	0.77	$\mathbf{0.52}$	$\mathbf{0.83}$	$\mathbf{0.81}$	$0.60_{\pm 0.03}$
	svhn	$0.80_{\pm 0.01}$	0.96	$0.55_{\pm 0.01}$	$\mathbf{0.83_{\pm 0.01}}$	0.96	$\mathbf{0.50_{\pm 0.01}}$
	tiny imagenet	0.82	0.79	$\mathbf{0.53}$	0.82	0.79	$0.58_{\pm 0.01}$
	uniform	$0.51_{\pm 0.09}$	$0.66_{\pm 0.05}$	$0.55_{\pm 0.09}$	$\mathbf{1.00}$	$\mathbf{1.00}$	$\mathbf{0.00}$
VGG16BN	bernoulli	$0.87_{\pm 0.02}$	$0.88_{\pm 0.02}$	$0.24_{\pm 0.03}$	$\mathbf{1.00}$	$\mathbf{1.00}$	$\mathbf{0.00}$
	blobs	0.95	0.97	$0.16_{\pm 0.01}$	$\mathbf{0.97}$	$\mathbf{0.98}$	$\mathbf{0.12_{\pm 0.02}}$
	cifar10	0.78	0.73	0.63	0.78	$\mathbf{0.74}$	$0.64_{\pm 0.01}$
	dtd	0.73	0.53	$0.62_{\pm 0.01}$	$\mathbf{0.86}$	$\mathbf{0.80_{\pm 0.01}}$	$\mathbf{0.50_{\pm 0.01}}$
	gaussian	$0.87_{\pm 0.02}$	$0.90_{\pm 0.01}$	$0.35_{\pm 0.04}$	$\mathbf{0.95_{\pm 0.01}}$	$\mathbf{0.97_{\pm 0.01}}$	$\mathbf{0.15_{\pm 0.03}}$
	lsun	0.85	0.82	0.47	$\mathbf{0.90}$	$\mathbf{0.89}$	$\mathbf{0.40}$
	places	0.82	0.78	0.55	$\mathbf{0.86}$	$\mathbf{0.85}$	$\mathbf{0.51}$
	svhn	$0.76_{\pm 0.01}$	0.95	$0.67_{\pm 0.02}$	$0.76_{\pm 0.01}$	0.95	$0.72_{\pm 0.04}$
	tiny imagenet	0.81	0.78	0.56	$\mathbf{0.83}$	$\mathbf{0.79}$	$\mathbf{0.55}$
	uniform	$0.86_{\pm 0.02}$	$0.89_{\pm 0.02}$	$0.28_{\pm 0.04}$	$\mathbf{1.00}$	$\mathbf{1.00}$	$\mathbf{0.00}$
WideResNet28x10	bernoulli	$0.97_{\pm 0.02}$	$0.96_{\pm 0.02}$	$0.04_{\pm 0.02}$	$\mathbf{1.00}$	$\mathbf{1.00}$	$\mathbf{0.00}$
	blobs	0.95	0.97	$0.16_{\pm 0.01}$	$\mathbf{0.98_{\pm 0.01}}$	$\mathbf{0.99}$	$\mathbf{0.09_{\pm 0.02}}$
	cifar10	0.80	0.74	0.53	$\mathbf{0.81}$	$\mathbf{0.76}$	$0.54_{\pm 0.01}$
	dtd	$0.84_{\pm 0.01}$	$0.75_{\pm 0.01}$	$0.54_{\pm 0.01}$	$\mathbf{0.88_{\pm 0.01}}$	$\mathbf{0.84_{\pm 0.01}}$	$\mathbf{0.49_{\pm 0.02}}$
	gaussian	$0.72_{\pm 0.08}$	$0.78_{\pm 0.05}$	$0.39_{\pm 0.09}$	$\mathbf{0.94_{\pm 0.03}}$	$\mathbf{0.97_{\pm 0.01}}$	$\mathbf{0.20_{\pm 0.07}}$
	lsun	0.87	$0.81_{\pm 0.01}$	0.35	$\mathbf{0.91}$	$\mathbf{0.90_{\pm 0.01}}$	$\mathbf{0.32_{\pm 0.01}}$
	places	0.84	$0.79_{\pm 0.01}$	$0.44_{\pm 0.01}$	$\mathbf{0.88}$	$\mathbf{0.87_{\pm 0.01}}$	$\mathbf{0.41_{\pm 0.01}}$
	svhn	0.80	0.95	$0.52_{\pm 0.01}$	$0.80_{\pm 0.01}$	0.95	$0.52_{\pm 0.01}$
	tiny imagenet	0.84	0.78	0.46	$\mathbf{0.85}$	$\mathbf{0.81}$	0.46
	uniform	$0.95_{\pm 0.01}$	$0.96_{\pm 0.01}$	$0.12_{\pm 0.03}$	$\mathbf{1.00}$	$\mathbf{1.00}$	$\mathbf{0.00}$

Table 7: SVHN results (averaged across 5 seeds)

Architecture	OOD dataset	DE			Ours		
		AUC (↑)	**AP (↑)**	**FPR95 (↓)**	**AUC (↑)**	**AP (↑)**	**FPR95 (↓)**
PreResNet164	bernoulli	1.00	0.99	0.01	1.00	**1.00**	**0.00**
	blobs	**1.00**	**0.99**	**0.01**	0.99	0.98	0.02
	cifar10	0.99	0.97	0.02	0.99	0.97	0.02
	cifar100	0.99	0.96	**0.02**	0.99	0.96	0.04
	dtd	0.99	0.94	0.02	**1.00**	**0.98**	**0.01**
	gaussian	1.00	0.99	0.01	1.00	**1.00**	**0.00**
	lsun	0.99	0.96	0.02	**1.00**	**0.98**	**0.01**
	places	0.99	0.96	0.02	**1.00**	**0.98**	**0.01**
	tiny imagenet	0.99	0.96	0.02	**1.00**	**0.97**	0.02
	uniform	1.00	0.99	0.01	1.00	**1.00**	**0.00**
VGG16BN	bernoulli	1.00	0.99	0.01	1.00	**1.00**	**0.00**
	blobs	1.00	0.98	0.01	1.00	**0.99**	0.01
	cifar10	0.99	0.95	**0.02**	0.99	**0.96**	0.03
	cifar100	0.99	0.94	**0.02**	0.99	**0.95**	0.05
	dtd	0.99	0.93	0.02	**1.00**	**0.97**	**0.01**
	gaussian	1.00	0.99	0.01	1.00	**1.00**	**0.00**
	lsun	0.99	0.95	0.02	**1.00**	**0.99**	**0.01**
	places	0.99	0.96	0.02	**1.00**	**0.98**	**0.01**
	tiny imagenet	0.99	0.95	**0.02**	0.99	**0.97**	0.03
	uniform	1.00	0.99	0.01	1.00	**1.00**	**0.00**
WideResNet28x10	bernoulli	1.00	$0.99_{\pm 0.01}$	$0.02_{\pm 0.01}$	1.00	1.00	**0.00**
	blobs	0.99	0.98	0.02	**1.00**	**0.99**	**0.01**
	cifar10	0.99	0.96	0.02	**1.00**	**0.97**	0.02
	cifar100	0.99	0.95	0.03	0.99	**0.97**	**0.02**
	dtd	0.99	$0.91_{\pm 0.01}$	0.04	**1.00**	**0.99**	**0.00**
	gaussian	1.00	0.98	0.02	1.00	**1.00**	**0.00**
	lsun	0.99	0.95	0.03	**1.00**	**0.99**	**0.00**
	places	0.99	0.95	0.03	**1.00**	**0.99**	**0.00**
	tiny imagenet	0.99	0.96	0.02	**1.00**	**0.98**	**0.01**
	uniform	0.99	$0.98_{\pm 0.01}$	$0.03_{\pm 0.01}$	**1.00**	**1.00**	**0.00**