Feature Space Particle Inference for Neural Network Ensembles

Shingo Yashima [1] **Teppei Suzuki** [1] **Kohta Ishikawa** [1] **Ikuro Sato** [1 2] **Rei Kawakami** [1 2]

Abstract

Ensembles of deep neural networks demonstrate improved performance over single models. For enhancing the diversity of ensemble members while keeping their performance, particle-based inference methods offer a promising approach from a Bayesian perspective. However, the best way to apply these methods to neural networks is still unclear: seeking samples from the weight-space posterior suffers from inefficiency due to the over-parameterization issues, while seeking samples directly from the function-space posterior often results in serious underfitting. In this study, we propose optimizing particles in the feature space where the activation of a specific intermediate layer lies to address the above-mentioned difficulties. Our method encourages each member to capture distinct features, which is expected to improve ensemble prediction robustness. Extensive evaluation on real-world datasets shows that our model significantly outperforms the gold-standard Deep Ensembles on various metrics, including accuracy, calibration, and robustness.

1. Introduction

Averaging predictions of multiple trained models has been a very popular technique in machine learning because it can significantly improve the generalization ability of a prediction system. In particular, ensemble methods of neural networks have recently achieved significant success in terms of predictive performance (Lakshminarayanan et al., 2017), uncertainty estimation (Ovadia et al., 2019), and robustness to adversarial attacks (Pang et al., 2019) or perturbations (Hendrycks & Dietterich, 2019). Among many ensemble methods, Deep Ensembles (Lakshminarayanan et al., 2017), which train each model independently from randomly initialized weights, has been the de facto approach with a decent performance and ease of implementation. However, it relies only on the randomness of initialization to generate different ensemble members and thus does not explicitly encourage diversity among them, which can cause redundancy in model averaging (Rame & Cord, 2021).

More recently, particle-based variational inference methods (Liu & Wang, 2016) have provided a promising approach for composing better ensembles from a Bayesian perspective (Wang et al., 2019; D'Angelo & Fortuin, 2021). These methods intend to approximate the Bayes posterior by the (pre-specified) finite number of models (called *particles*) using deterministic optimization. They have greater non-parametric flexibility than classical variational inferences (Blundell et al., 2015; Gal & Ghahramani, 2016) and provide better efficiency than sampling-based Markov chain Monte Carlo (MCMC) methods (Neal, 1996; Welling & Teh, 2011). Notably, their optimizations take interactions between models into account using kernel functions and explicitly promote model diversity, unlike Deep Ensembles.

These inferences are usually performed on the weight space of neural networks to approximate weight-space posterior. However, this is far from ideal due to the over-parameterized nature of recent neural networks. Such models have many local modes in the weight-space posterior that are distant from each other, yet corresponding to the same predictive function (Fort et al., 2019; Entezari et al., 2021). Therefore, promoting diversity of weights does not necessarily result in diversity as a predictive function, and it can yield a degenerate ensemble when the number of models is limited.

To circumvent this, Wang et al. (2019) proposed performing the inference on model outputs to obtain an approximation of the function-space posterior. They treat output logits on data points as inferred parameters and promote diversity on them. Although they do not suffer from the above-mentioned degeneration issues, directly seeking a posterior on the output space of neural network functions often results in severe underfitting (D'Angelo & Fortuin, 2021). Overall, none of these methods have shown significant improvements over naive Deep Ensembles in accuracy and calibration (D'Angelo et al., 2021).

On the other hand, a recent theoretical study suggests that the critical component of the success of neural network ensembles is the *multi-view* structure of data (Allen-Zhu

[1] Denso IT Laboratory Inc., Tokyo, Japan [2] Tokyo Institute of Technology, Tokyo, Japan. Correspondence to: Shingo Yashima <yashima.shingo@core.d-itlab.co.jp>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

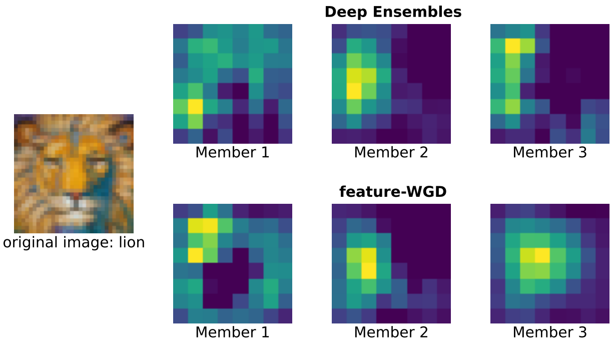

Figure 1. Class activation maps for an ensemble of WRN-28-2 on CIFAR-100 trained with Deep Ensembles (up) and the proposed method (bottom). The proposed method captures more diverse features (mane, face) than Deep Ensembles. Use Grad-CAM (Selvaraju et al., 2017) for visualization.

& Li, 2020). They claimed that when data have multiple features that can be used to classify them correctly (which they call multi-view), and each member of an ensemble captures different features from each other, the prediction performance on the test data can be boosted by ensembling. This multi-view structure is typical in real-world data: for example, the image of a male lion presented in Figure 1 can be classified correctly by looking either at its mane or face. Intuitively, if we ensemble models which look at different parts of a lion to classify, the robust prediction would be possible even on, for example, female lions without a mane.

Leveraging these works, we hypothesize that promoting the diversity of feature extractors rather than predictive functions encourages each model to capture different data views, resulting in better performance of an ensemble. We expect that feature diversity allows the same predictive functions on easy data, which prevents underfitting, and yields different predictive functions on hard data, which increases robustness. To this end, we propose a framework for performing particle-based variational inference on the feature space of networks. Our contributions are summarized as follows:

- We formalize particle-based inference methods on feature space of neural networks (Section 3), which encourages each network to capture distinct features to classify data. In our construction, multiple feature extractors are connected to a common classification layer, so that the feature distribution from all feature extractors is adequately controlled with respect to the subsequent layer. It does not suffer from overparameterization issues in weight-space inference, and it is hard to underfit, unlike function-space inference.
- We performed extensive experiments to evaluate the classification and calibration ability of the proposed approach on typical image datasets, including CIFAR and ImageNet (Section 4). The results show the superiority of feature-space inference over weight or function-space inference. Moreover, the proposed approach consistently outperforms gold-standard Deep Ensembles in generalization ability and robustness.

2. Background

2.1. Bayesian neural networks and Deep Ensembles

In typical supervised deep learning, we consider a likelihood function $p(y|f(x;w))$ with a neural network f parameterized by w. Then we maximize the likelihood of training data $\mathcal{D} = \{(x_b, y_b)\}_b$ with respect to w to obtain a network that adequately explains the data. In Bayesian neural networks (BNNs), we intend to ensemble all-likely networks which explain training data $\mathcal{D}$ by drawing weights from the posterior distribution $p(w|\mathcal{D}) \propto p(w) \prod_{(x,y)\in\mathcal{D}} p(y|f(x;w))$, where $p(w)$ is a prior over weights. When making a prediction on a test point x_*, we marginalize the predictive functions over the posterior:

$$p(y_*|x_*, \mathcal{D}) = \int_{\mathcal{W}} p(y_*|f(x_*;w))p(w|\mathcal{D})\mathrm{d}w \qquad (1)$$

The exact posterior is generally intractable in the case of neural networks; thus, various approximation methods for BNN have been developed so far, including variational inference methods (Graves, 2011; Blundell et al., 2015; Gal & Ghahramani, 2016; Wen et al., 2018) and MCMC methods (Neal, 1996; Welling & Teh, 2011). With weights $\{w_i\}_{i=1}^n$ sampled from the (possibly approximated) posterior, the predictive distribution is obtained by Monte Carlo estimates of (1): $p(y_*|x_*, \mathcal{D}) \approx \frac{1}{n}\sum_{i=1}^n p(y_*|f(x_*;w_i))$.

On the other hand, Deep Ensembles (Lakshminarayanan et al., 2017) compose an ensemble of networks by initializing and training each network independently. Although they do not explicitly sample network parameters from particular distributions like BNNs, diverse networks which explain data can be obtained, because different initial values of each network weights result in different solutions in non-convex neural network training (Fort et al., 2019).

From a practical viewpoint, it is known that BNNs are generally inferior to Deep Ensembles in terms of both accuracy and uncertainty estimation (Ashukha et al., 2020; Ovadia et al., 2019; Gustafsson et al., 2020). This is because the expressive power of the approximated posterior in practical BNNs is not sufficient to capture various modes in the complex weight-space posterior. As a result, each member in a BNN ensemble is less diverse as a predictive function than that in Deep Ensembles (Fort et al., 2019).

2.2. Particle-based variational inference for BNNs

Recently, particle-based variational inference methods (Liu & Wang, 2016; Chen et al., 2018) have received attention for developing better ensemble methods in BNNs. These methods seek to approximate the target posterior $p(w|\mathcal{D})$ with the pre-specified number of particles $\{w_i\}_{i=1}^n$ by transporting them using deterministic optimization. Its non-parametric flexibility allows it to capture more complex posterior than traditional variational inference methods. In addition, the interaction between particles in optimization enables a more particle-efficient approximation of the posterior than sampling-based MCMC.

In typical particle-based inference methods, such as Stein variational gradient descent (SVGD) (Liu & Wang, 2016), an update direction v for each particle $\{w_i\}_{i=1}^n$ is described in the following form:

$$v(w_i) = \sum_{j=1}^{n} \beta_{ij} \nabla \log p(w_j|\mathcal{D}) + \gamma_{ij} \nabla_{w_j} k(w_i, w_j)$$

where k is a positive definite kernel and β_{ij} and γ_{ij} are scalars that depend on $\{w_i\}_{i=1}^n$. As we can see, the update direction v consists of two parts: the *driving term*, which pushes particles towards high-density regions in the posterior, and the *repulsive term*, which prevents particles from collapsing into a single MAP estimate. Thus, their training procedure can be viewed as Deep Ensembles with repulsive forces between members (D'Angelo & Fortuin, 2021). They explicitly promote diversity through member interactions and, therefore, can be expected to produce a better performing ensemble than naive Deep Ensembles.

However, when applied to over-parameterized models such as neural networks, they can produce degenerate ensemble members when the number of particles is limited, whereas a consistency to the true posterior is guaranteed in many particle limits (Liu, 2017). That is, because different weights far from each other can map to the same function in such over-parameterized models, the repulsive term on weights does not effectively promote diversity as a predictive function.

Regarding this, Wang et al. (2019) proposed to directly seek a functional posterior $p(f|\mathcal{D}) \propto p(f) \prod_{(x,y)\in\mathcal{D}} p(y|f(x))$ by performing inference on the output logits of a network evaluated on data points. However, it often shows severe underfitting in real-world image classification tasks (D'Angelo et al., 2021). The reason behind this is still not apparent, but one possible explanation is the dangers of directly promoting diversity in logit space. In typical image datasets such as CIFAR, a label is almost deterministically produced for a given image (i.e., there is low label noise (Tsybakov, 2004)). Thus, there is no room for diversity in the logit space for most images in such datasets. More recently, D'Angelo et al. (2021) proposed a hybrid use of weight-space and function-space update rules. They showed an improved performance using additional stochasticity on gradients (Gallego & Insua, 2018), but Deep Ensembles still perform the best in the deterministic setting in terms of accuracy and negative log-likelihood.

3. Particle-based Inference on Feature Space

In the following, we present a framework for feature space particle-based inference. Its advantages over weight or function-space inference are summarized as follows:

- Given a shared classifier on the top of independent feature extractors as described in follows, a posterior of feature extractors is considered to be relatively simple and hard to suffer from the over-parameterization problem observed in weight-space inference.
- It does not directly promote diversity on output logits, even allowing the same prediction results for easy data. This is expected to prevent the underfitting observed in the function-space inference.
- By encouraging each feature extractor to capture different features to classify the same input data, it can exploit the multi-view structure of data, which is considered a critical component of the performance gain by ensembling.

We begin by introducing a specific form of the inference method used in this study and then show how to perform inference on feature extractors. Finally, we discuss the prior selection of feature extractors and the computational efficiency of our algorithm.

3.1. Wasserstein gradient descent

There are various formulations of particle-based variational inference, depending on how variational approximation and discretization are applied to derive finite-particle update rules. This section introduces a specific form of the inference method used in this study, named Wasserstein gradient descent (WGD) (Liu et al., 2019; Wang et al., 2021; D'Angelo & Fortuin, 2021). Note that the choice of an inference method is independent of the space in which the inference is performed, and we denote the variables to be inferred as $w \in \mathcal{W}$ here.

Given the target posterior distribution $p(\cdot|\mathcal{D})$, our final objective is to minimize the following KL divergence with respect to the current particle distribution q:

$$\mathrm{KL}_{p(\cdot|\mathcal{D})}(q) = \int_{\mathcal{W}} (\log p(w|\mathcal{D}) - \log q(w)) p(w|\mathcal{D}) \mathrm{d}w.$$

Particle-based inference can be formulated as a gradient descent optimization of the above functional in the Wasserstein space $\mathcal{P}_2(\mathcal{W})$ equipped with the well-known Wasserstein distance W_2 (Ambrosio et al., 2008; Villani, 2009). Specifically, its Wasserstein gradient flow $\{(q_t)_t\}$, which is roughly the family of steepest descending curves for $\mathrm{KL}_{p(\cdot|\mathcal{D})}$ in $\mathcal{P}_2(\mathcal{W})$, has its tangent vector v_t at any t being:

$$v_t(w) = \nabla \log p(w|\mathcal{D}) - \nabla \log q_t(w) \tag{2}$$

whenever q_t is absolutely continuous (Liu et al., 2019). Intuitively, $v_t(w)$ represents the direction in which the probability mass on the point w of q_t should be moved in order to bring q_t close to p.

Although the first term of (2) can be calculated using an unnormalized posterior, we do not have access to the analytical form of the second term $\nabla \log q_t$ when performing gradient descent along with v_t using finite particles $\{w_i^t\}_i$. In WGD, we consider using kernel density estimation (KDE) to approximate q_t with $\{w_i^t\}_i$: $q_t(w) \propto \sum_{j=1}^n k(w, w_j^t)$. Here k is a given positive definite kernel function such as RBF kernel. Then, an approximation of the second term in (2) is given by

$$\nabla \log q_t(w_i^t) \approx \frac{\sum_{j=1}^n \nabla_{w_i^t} k(w_i^t, w_j^t)}{\sum_{j=1}^n k(w_i^t, w_j^t)}.$$

Using the above formula, the update rule of particle w_i^t is obtained as follows:

$$\begin{aligned} w_i^{t+1} &= w_i^t + \alpha_t v_t(w_i^t) \\ &\approx w_i^t + \alpha_t \left(\nabla \log p(w_i^t|\mathcal{D}) - \frac{\sum_{j=1}^n \nabla_{w_i^t} k(w_i^t, w_j^t)}{\sum_{j=1}^n k(w_i^t, w_j^t)} \right), \end{aligned} \tag{3}$$

where $\alpha_t > 0$ is a step size parameter for iteration t. Unlike SVGD, the update rule of WGD does not include the averaging of the gradient of the log posterior between particles, which is known to be harmful in high-dimensional settings such as neural networks (D'Angelo & Fortuin, 2021). We note that, although several studies proposed more particle-efficient update rules than WGD (Li & Turner, 2018; Shi et al., 2018), we do not go deep into a choice of inference algorithm itself and use WGD as a simple baseline.

3.2. WGD on feature space

Let $h(\cdot; w)$ be a feature extractor and $c(\cdot; \theta)$ be a classifier of the neural network f parametrized by w and θ, respectively: $f(\cdot; w, \theta) = c(\cdot; \theta) \circ h(\cdot; w)$. In typical image classification networks like ResNet (He et al., 2016), c corresponds to a final linear layer and h corresponds to a whole network before that. We consider each ensemble member to have an independent feature extractor $h(\cdot; w_i)$ and a shared classifier $c(\cdot; \theta)$; therefore, i-th member of an ensemble is written as $f(\cdot; w_i, \theta) = c(\cdot; \theta) \circ h(\cdot; w_i)$. The classifier sharing is essential for our formulation: by doing this, the output space of each feature extractor does not suffer from permutations of the subsequent classifier and thus is expected to share same semantic information, which enables performing particle inference on feature extractors as described below.

In this study, we consider a shared classifier $c(\cdot; \theta)$ as a deterministic function, whose parameters are not inferred in particle optimization. Thus, a posterior distribution that we seek to approximate by particles is given by

$$\begin{aligned} p(h|\mathcal{D}) &\propto p(h) p(\mathcal{D}|c(\cdot;\theta) \circ h) \\ &= p(h) \prod_{(x,y)\in\mathcal{D}} p\left(y|c(h(x);\theta)\right). \end{aligned}$$

However, obtaining a functional posterior of h is neither tractable nor practical in the case of neural networks. Following (Wang et al., 2019; D'Angelo & Fortuin, 2021), we instead perform an inference on feature values evaluated at training points $\mathcal{X} = \{x_b\}_b$: $\mathbf{h} = \mathrm{vec}\left(\{h(x)\}_{x\in\mathcal{X}}\right)$. Plugging into the update rule of WGD (3), the update direction of $\mathbf{h}_i^t$ (i-th particle on iteration t) is written as

$$v_t(\mathbf{h}_i^t) = \nabla_{\mathbf{h}_i^t} \log p(\mathbf{h}_i^t|\mathcal{D}) - \frac{\sum_{j=1}^n \nabla_{\mathbf{h}_i^t} k(\mathbf{h}_i^t, \mathbf{h}_j^t)}{\sum_{j=1}^n k(\mathbf{h}_i^t, \mathbf{h}_j^t)}. \tag{4}$$

The second term is the repulsive force, which encourages the features of each member to be different from each other, whereas the first term promotes them to correctly classify data. The first term, the log-gradient of the posterior, is decomposed as follows:

$$\nabla_{\mathbf{h}_i^t} \log p(\mathbf{h}_i^t|\mathcal{D}) = \nabla_{\mathbf{h}_i^t} \log p(\mathbf{h}_i^t) + \nabla_{\mathbf{h}_i^t} \log p(\mathcal{D}|\mathbf{h}_i^t).$$

The first term is a log-gradient of the prior, and the prior choice is discussed afterward. The second term corresponds to a log-gradient of the data likelihood with respect to the feature values, given a shared classifier:

$$\nabla_{\mathbf{h}_i^t} \log p(\mathcal{D}|\mathbf{h}_i^t) = \nabla_{\mathbf{h}_i^t} \sum_{(x,y)\in\mathcal{D}} \log p(y|c(h(x; w_i^t); \theta)).$$

In the training procedure, we transport the feature extractors $\{\mathbf{h}_i\}_{i=1}^n$ by updating the weights $\{w_i\}_{i=1}^n$. An update rule of weights can be obtained by projecting the functional update (4) back to the weight space using a Jacobian:

$$w_i^{t+1} = w_i^t + \frac{\alpha_t}{|\mathcal{D}|} \left(\frac{\partial \mathbf{h}_i^t}{\partial w_i^t} \right)^\top v_t(\mathbf{h}_i^t). \tag{5}$$

This update can be implemented using standard backpropagation. In addition to the particle optimization in the feature space, we simultaneously update the shared classifier $c(\cdot; \theta)$ by maximizing the average of log-likelihood over the particles and data:

$$\theta^{t+1} = \theta^t + \frac{\alpha_t}{n|\mathcal{D}|} \sum_{i=1}^n \sum_{(x,y)\in\mathcal{D}} \nabla_{\theta^t} \log p\left(y \middle| c(h(x; w_i^t); \theta^t)\right). \tag{6}$$

In practice, the update (5) and (6) can be performed in a mini-batch manner by modifying the feature values $\mathbf{h}_i^t$ to that evaluated on a current mini-batch. In addition, weight decay is applied to these parameter updates to prevent overfitting. For making a prediction on a test point x_* after training, we approximate the predictive distribution (1) using the obtained weights $\{w_i\}_{i=1}^n$ and θ:

$$p(y_*|x_*, \mathcal{D}) \approx \frac{1}{n}\sum_{i=1}^{n} p(y_*|f(x_*; w_i, \theta)).$$

Projection in the repulsive term. Although the overparameterization issue is circumvented, the inferred parameter $\mathbf{h}$ is generally high-dimensional, which causes inefficient sampling from the posterior. In particular, the repulsive term of the update rule (4) involves KDE, which is known to suffer from the curse of dimensionality (Scott, 1991). Thus, inspired by (Wang et al., 2021; Chen & Ghattas, 2020), we consider estimating the density in a low-dimensional subspace in which the likelihood of data changes significantly. To find such a subspace, we use the gradient information of the log-likelihood by defining a matrix H_t as

$$H_t = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{h}_i^t} \log p(\mathcal{D}|\mathbf{h}_i^t)\left(\nabla_{\mathbf{h}_i^t} \log p(\mathcal{D}|\mathbf{h}_i^t)\right)^\top \quad (7)$$

and use a r-dimensional subspace spanned by $\Psi_r = (\psi_1, \ldots, \psi_r)$, where ψ_i is the i-th dominant eigenvector of H_t and $r \leq n$. Denoting $\mathbf{z}_i^t = \Psi_r^\top \mathbf{h}_i^t \in \mathbb{R}^r$, the projected version of the repulsive term in (4) is obtained as

$$\Psi_r \frac{\sum_{j=1}^n \nabla_{\mathbf{z}_i^t} k(\mathbf{z}_i^t, \mathbf{z}_j^t)}{\sum_{j=1}^n k(\mathbf{z}_i^t, \mathbf{z}_j^t)}. \quad (8)$$

Roughly speaking, we evaluate the repulsive term only on feature elements that significantly affect the prediction results. Note that in (Wang et al., 2021; Chen & Ghattas, 2020), the entire update rule including the driving term is projected onto a subspace, but we only project the repulsive (KDE) term because we found that yields more stable training on neural networks.

In addition, this projection can mitigate the risk of promoting a "useless" diversity of features. As in single model training, different feature dimensions of a trained ensemble may carry similar semantic information. As a result, the obtained diversity on such feature dimensions may not represent true semantic diversity. However, in such a case, there would be a subspace that does not influence prediction results. For example, when subsequent classifier weights on two feature elements are perfectly identical, increasing one feature value and decreasing the other does not affect the output logits. Such a redundant subspace is likely to be removed by the projection because we compose subspace by adopting directions that strongly influence prediction results, as in (7).

We denote the WGD implemented in each space as $\{$weight, function, feature$\}$-WGD, respectively. The entire inference procedure is summarized in Appendix A.

3.3. Prior choice

How to impose a functional prior on the feature extractor h (or $\mathbf{h}$) is a major concern. It can be defined either as a push-forward measure of the weight-space prior or a stochastic process (e.g., Gaussian process (GP)) configured regardless of the parametric form of a network. The former approach can be implemented using an empirical approximation using weight samples (Wang et al., 2019), or a gradient estimation of implicit distributions (Li & Turner, 2018; Shi et al., 2018), but requires additional calculation costs. Moreover, it has been shown that functional priors induced by standard Gaussian weight priors exhibit spiking behavior (Wenzel et al., 2020a; Tran et al., 2020), which can be problematic in training. For these reasons, we adopt the latter approach and impose independent and identical priors on each element of $\mathbf{h}$ for simplicity of implementation. Setting independent priors on features of different data points may seem weird at first, because it does not impose any functional smoothness of feature extractor h, unlike typical GP regression (Rasmussen et al., 2006). However, we expect that it still yields sufficient smooth functions owing to a well-known inductive bias in neural network training, leading to generalizing local minima (Neyshabur et al., 2015; Mandt et al., 2017). Further investigation on priors, mainly about the correlation between data (Wilson & Izmailov, 2020), is required in future work.

We consider three distributions for priors: normal, Cauchy, and uniform. Cauchy, which is known as a weakly informative prior, is often preferred for robustness as it places less density at the mean owing to the heavier tails (Gelman, 2006). We expect this to be more suitable for feature-space priors because the activation of a trained network often shows heavy-tailed distributions (Favaro et al., 2020). Uniform prior is improper, and imposing such a prior corresponds to removing the effect of priors in the update rule (4). We explore their practical performance in Section 4.3 and adopt Cauchy as a default prior for its superior performance.

3.4. Computational overhead

Compared to naive Deep Ensembles, we require additional computation to calculate the repulsive term between members, as shown in (8). Denoting the batch size as B and the feature dimension of each network as H, we need $O(n^2BH)$ time to calculate the projection basis Ψ_r by exact SVD. Assuming that the kernel evaluation is $O(r)$ (e.g., RBF), evaluating (8) takes $O(n^2r + rBH)$ time. On

Table 1. Results for Wide ResNet-16-4 on CIFAR-10 with an ensemble size of 10, evaluated over 5 seeds.

METHOD	ACCURACY(↑)	NLL(↓)	BRIER(↓)	ECE(↓)	cA / cNLL / cBRIER / cECE
SINGLE	95.4 ± 0.2	0.145 ± 0.006	0.069 ± 0.003	0.007 ± 0.000	73.7 / 0.796 / 0.349 / **0.020**
DEEP ENSEMBLES	96.4 ± 0.1	0.110 ± 0.001	0.054 ± 0.001	0.007 ± 0.000	76.7 / 0.698 / 0.310 / 0.025
WEIGHT-WGD	96.4 ± 0.1	0.111 ± 0.002	0.054 ± 0.001	0.007 ± 0.001	76.7 / 0.702 / 0.312 / 0.026
FUNCTION-WGD	96.1 ± 0.1	0.124 ± 0.001	0.059 ± 0.001	0.007 ± 0.001	75.7 / 0.736 / 0.322 / 0.024
FEATURE-WGD	**96.5 ± 0.1**	**0.107 ± 0.001**	**0.052 ± 0.001**	**0.006 ± 0.001**	**77.3 / 0.681 / 0.302 / 0.020**

Table 2. Results for Wide ResNet-16-4 on CIFAR-100 with an ensemble size of 10, evaluated over 5 seeds.

METHOD	ACCURACY(↑)	NLL(↓)	BRIER(↓)	ECE(↓)	cA / cNLL / cBRIER / cECE
SINGLE	77.4 ± 0.3	0.835 ± 0.007	0.316 ± 0.003	0.030 ± 0.003	46.7 / 2.279 / 0.658 / 0.035
DEEP ENSEMBLES	82.3 ± 0.2	0.632 ± 0.004	0.249 ± 0.001	0.020 ± 0.001	52.9 / 1.971 / 0.590 / 0.032
WEIGHT-WGD	82.3 ± 0.1	0.633 ± 0.002	0.250 ± 0.001	0.021 ± 0.001	52.8 / 1.967 / 0.589 / 0.031
FUNCTION-WGD	79.0 ± 0.1	0.715 ± 0.003	0.286 ± 0.001	0.018 ± 0.002	49.5 / 2.133 / 0.623 / 0.034
FEATURE-WGD	**82.9 ± 0.2**	**0.624 ± 0.002**	**0.243 ± 0.001**	**0.017 ± 0.001**	**53.5 / 1.955 / 0.584 / 0.029**

the other hand, the back-propagation, which both methods have in common, takes $O(nBD)$ (or $O(BD)$ with model parallelization), where D is the number of weights in the network. In typical classification models such as ResNet, $H \sim 10^3$, $B \sim 10^2$, and $D \sim 10^7$. Therefore, the overall computational cost is dominated by the back-propagation for a practical ensemble size $n \sim 10$. In our experiment on CIFAR-100 for Wide ResNet-16-4 with an ensemble size of 10, feature-WGD takes approximately 17s for one epoch, while Deep Ensembles take 16s on four A100 GPUs.

4. Experiments

In this section, we present the results on popular image classification tasks: CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and ImageNet (Deng et al., 2009). For CIFAR, we use a Wide ResNet-16-4 as the base architecture for its compactness and decent performance (Zagoruyko & Komodakis, 2016). For ImageNet, we use ResNet-50 as it is the most commonly benchmarked model (He et al., 2016). We follow the standard scheduling, augmentation, and regularization schemes in the literature (Chen et al., 2020), which are summarized in Appendix B. Note that these training schemes have been heavily tuned in prior works to prevent overfitting, so it is hard to improve ensemble performance simply by improving single model performance. In addition, it is worth mentioning that we use SGD with Nesterov acceleration as an optimizer, whereas Adam (Kingma & Ba, 2015) has been traditionally used for particle-based inference of neural networks (Liu & Wang, 2016; Wang et al., 2019; D'Angelo et al., 2021). We think this yields a more fair and practical comparison of particle-based inference methods and Deep Ensembles since SGD generally performs significantly better than Adam on these classification tasks (Wilson et al., 2017). For a kernel used in WGD, we adopt a standard RBF kernel and determine its bandwidth by the median heuristic (Schölkopf et al., 2002). Code is available at: `https://github.com/DensoITLab/featurePI`.

Metrics. We report the test accuracy, negative log-likelihood (NLL), Brier score (Brier, 1950), and the expected calibration error (ECE) (Naeini et al., 2015). For calibration metrics, we follow the procedure from (Ashukha et al., 2020), which evaluates the metrics after temperature scaling (Guo et al., 2017). In addition to the in-domain evaluation, we measure robustness to image perturbations by evaluating these metrics on the corrupted versions of these datasets (CIFAR-10-C, CIFAR-100-C, and ImageNet-C) (Hendrycks & Dietterich, 2019). They apply a set of 15 common visual corruptions with intensities ranging from 1 to 5. We average each metric over all corruption types and intensities (cA(ccuracy), cNLL, cBrier, and cECE).

4.1. Comparison of WGD on different space and Deep Ensembles

Firstly, we test our central hypothesis, which states that promoting feature space diversity improves ensemble performance. To this end, we compare WGD with varying inference space ({weight, function, feature}-WGD) and gold-standard Deep Ensembles, as well as a single model baseline. Except for the difference in the inference space and prior parameters determined by the standard cross-validation, we adopted the same training scheme for all methods.

CIFAR-10 and CIFAR-100. We train Wide ResNet-16-4 with an ensemble size of 10 on both datasets. The result presented in Table 1 (CIFAR-10) and Table 2 (CIFAR-100)

Table 3. Results for ResNet-50 on ImageNet with an ensemble size of 5. Note that we only evaluate 1 run due to the computational cost.

Method	Accuracy(↑)	NLL(↓)	Brier(↓)	ECE(↓)	cA / cNLL / cBrier / cECE
Single	75.7	0.954	0.338	0.018	37.7 / 3.235 / 0.738 / 0.021
Deep Ensembles	**78.0**	**0.853**	**0.309**	0.019	40.9 / 3.011 / 0.706 / **0.015**
feature-WGD	**78.0**	0.859	**0.309**	**0.015**	**42.4 / 2.923 / 0.693** / 0.018

show that feature-WGD performs the best in terms of accuracy and calibration metrics on both the in-domain and the corrupted test set. For in-domain evaluation, the improvement of feature-WGD over Deep Ensembles is relatively small on CIFAR-10, because a single model already performs relatively well. On the other hand, a significant performance gain (e.g., +0.6% in accuracy) for the more difficult CIFAR-100 is observed. For corrupted data, we observe much better accuracy and calibration of feature-WGD on both datasets. This highlights the robustness induced by the exploitation of various data views.

The performance of weight-WGD almost coincides with that of Deep Ensembles, indicating that the weight-space repulsive term cannot exploit diversity beyond the difference in initialization. As reported in previous works (D'Angelo & Fortuin, 2021; D'Angelo et al., 2021), function-WGD shows a severe underfitting even in the optimized prior, which can be attributed to the harmfulness of the function-space repulsive term on these datasets. We think this is clear evidence of the advantages of feature-space inference over weight or function-space inference.

ImageNet. We train ResNet-50 with an ensemble size of 5. Here we only evaluate feature-WGD and Deep Ensembles. The results are presented in Table 3. Although the in-domain accuracy and calibration scores are almost comparable, robustness to corruption is clearly improved (e.g., +1.5% in accuracy) by feature-WGD. Figure 2 examines the performance under corruption in more detail by plotting results across corruption types for each corruption severity. Feature-WGD shows better robustness than Deep Ensembles even in the intense corruption.

4.2. Comparison with SOTA ensemble methods

Here, we compare feature-WGD with other ensemble methods proposed in the literature from a practical viewpoint. We select ADP (Pang et al., 2019) and DICE (Rame & Cord, 2021) as both report superior performance to Deep Ensembles. Note that we do not compare with Bayesian methods here, because generally they have not shown superior performance to Deep Ensembles with the same ensemble size (Ashukha et al., 2020). Due to code availability, we compare the reported values for ResNet-32 and Wide ResNet-28-2 on CIFAR-100 in (Rame & Cord, 2021) and follow the same training procedure for feature-WGD.

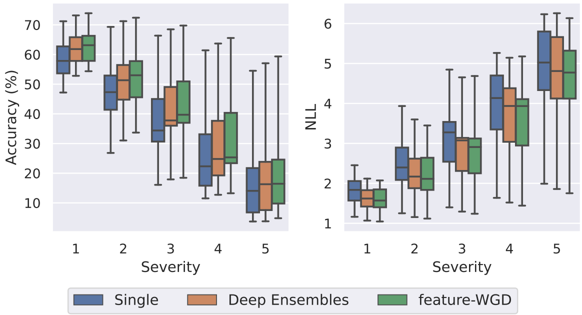

Figure 2. Results for ResNet-50 on ImageNet-C with an ensemble size of 5. We plot accuracy and NLL evaluated on 15 corruption types with corruption severity ranging 1-5.

The results are presented in Table 4. For Wide ResNet-28-2 with an ensemble size of 3, feature-WGD performs the best, as is DICE. For ResNet-32 with an ensemble size of 4, feature-WGD outperforms Deep Ensembles and ADP but falls short of DICE. We think one possible explanation for this is the insufficient width of the networks, as discussed later in Section 4.3. We note that DICE requires additional networks to estimate mutual information between models and adversarial training on them, whereas feature-WGD only modifies the gradient calculation in standard training. Overall, these results suggest that our method works well as a practical ensemble method, even though those ensemble size is rather small for a Bayesian method.

Table 4. Comparison with SOTA ensemble methods on CIFAR-100. We include the values reported in (Rame & Cord, 2021). Evaluated over 5 seeds.

Method	ResNet-32×4	WRN-28-2×3
Deep Ensembles	77.4 ± 0.1	80.0 ± 0.2
ADP	77.5 ± 0.3	80.0 ± 0.2
DICE	$\mathbf{77.9 \pm 0.1}$	$\mathbf{80.6 \pm 0.1}$
feature-WGD	77.6 ± 0.2	$\mathbf{80.6 \pm 0.2}$

4.3. In-depth experiments

This section delves deeper into the nature of our feature space particle inference and presents some guidelines about the choice of prior and base architecture for practical implementations.

Quantitative evaluation of feature diversity. Here we investigate how diverse the features obtained in feature-WGD are when compared to Deep Ensembles. We examine the diversity of class activation maps created by Grad-CAM (Selvaraju et al., 2017) instead of raw feature values because feature space is not shared among members in Deep Ensembles. Specifically, we compute an inner product of l_2-normalized class activation maps of an input image between two ensemble members as a similarity measure and average it across all pairs in an ensemble and test images. The results in Figure 3 show that the similarity of feature-WGD is consistently lower than that of Deep Ensembles across datasets used in Section 4.1, showing that members in the feature-WGD ensemble look at more various parts of images from each other. Furthermore, a larger drop in similarity is observed in more complex, multi-object datasets (ImageNet) than in less complicated, object-centric datasets (CIFAR-10). For corrupted datasets, where some features might be unavailable to classify data, we observe the similarity of both methods decreases from non-corrupted datasets, and feature-WGD further decreases it. From this and results in Section 4.1, we can see that feature-WGD indeed exploits the multi-view structure of data and achieves higher classification robustness.

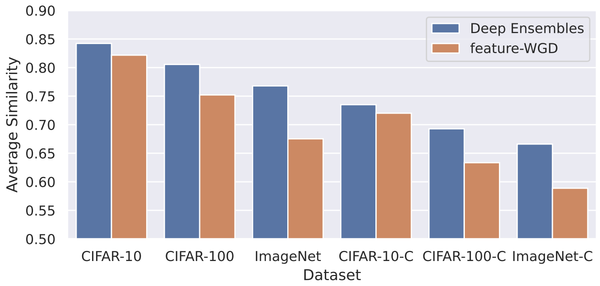

Figure 3. Average similarities of class activation maps produced by Grad-CAM among ensemble members of Deep Ensembles and feature-WGD on CIFAR-10, CIFAR-100, ImageNet, and those corrupted counterparts.

Prior choice. To examine the practical choice of priors in feature-WGD, we experiment with our CIFAR-100 setup for three priors: normal, Cauchy, and uniform. Prior parameters are optimized by cross-validation. The results in Table 5 show that Cauchy performs the best both in terms of accuracy and calibration, which may be attributed to the heavy-tail nature of neural network features, as discussed in Section 3.3. The uniform prior performs worse than Deep Ensembles, indicating that some informative priors are necessary for feature-WGD. The harmfulness of uniform priors has been suggested in the Bayesian statistics literature (Gelman & Yao, 2020).

Table 5. Results for Wide ResNet-16-4 on CIFAR-100 with an ensemble size of 10 trained with feature-WGD on different priors. Averaged over 5 seeds.

PRIOR	ACCURACY	NLL	BRIER	ECE
NORMAL	82.7	0.628	0.245	**0.017**
CAUCHY	**82.9**	**0.624**	**0.243**	**0.017**
UNIFORM	82.1	0.634	0.251	0.020

Improvement in larger ensemble size. Thus far, feature-WGD demonstrates better performance than Deep Ensembles with the same ensemble size. Then the natural question is, *when increasing an ensemble size, does the performance of Deep Ensembles catch up with feature-WGD?* Figure 4 shows test accuracy and negative log-likelihood of these methods when increasing an ensemble size up to 20. We can observe Deep Ensembles face saturation in both metrics, whereas feature-WGD improves its performance even in large ensemble sizes. This possibly indicates the limitations of relying only on differences in initial values to find a variety of solutions and the benefit of explicitly promoting feature diversity.

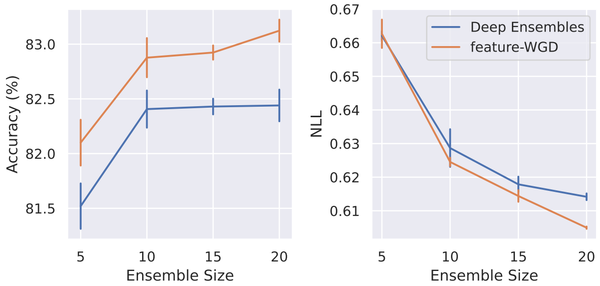

Figure 4. Accuracies (left) and NLLs (right) for Wide ResNet-16-4 on CIFAR-100 in the large ensemble sizes, evaluated over 3 seeds.

Need for network width. As we promote diversity in shared feature space, we expect feature-WGD needs more redundant feature capacity than standard training to exploit multi-view structures. Table 6 shows the accuracy results of feature-WGD and Deep Ensembles for our CIFAR-100 setting with varying network width factors. As expected, the improvement of accuracy by feature-WGD is relatively small in narrow networks (width factor 2) compared to wider networks (width factors 4 and 8). We believe this is one reason for the relatively poor performance of feature-WGD for ResNet-32 in Table 4, which has only 64 feature dimensions to classify 100 classes.

Table 6. Accuracy improvement of feature-WGD over Deep Ensembles on CIFAR-100 with an ensemble size of 10 for different network width factors (Wide ResNet-16-{2, 4, 8}). Averaged over 3 seeds.

WIDTH FACTOR	2	4	8
DEEP ENSEMBLES	80.1	82.3	83.6
FEATURE-WGD	80.3	82.9	84.1
IMPROVEMENT	+0.2	+0.6	+0.5

5. Related Work

In addition to related works addressed in Section 2, we highlight some relevant studies on ensembles and feature learning of neural networks.

Bayesian neural networks. Traditional MCMC methods (MacKay, 1992; Neal, 1996) are considered as the gold standard for BNNs (Wenzel et al., 2020a), but face difficulties in modern large-scale learning. Many practical BNNs are proposed, including variational inference (Kingma et al., 2015; Wen et al., 2018), K-FAC Laplace approximation (Ritter et al., 2018), SWAG (Maddox et al., 2019), subspace inference (Izmailov et al., 2020), and SG-MCMC (Welling & Teh, 2011; Zhang et al., 2020b). These works enable Bayesian inference on large problems, but they still fall short of Deep Ensembles in terms of both accuracy and uncertainty estimation (Ashukha et al., 2020).

Non-Bayesian ensemble methods. Beyond random initialization (Lakshminarayanan et al., 2017), some studies apply different augmentations (Dvornik et al., 2019) or hyperparameters (Wenzel et al., 2020b) to improve the diversity, but they are highly dependent on domain knowledge and engineering. For methods that explicitly diversify network outputs, ADP (Pang et al., 2019) promotes the orthogonality of the non-maximal predictions. DICE (Rame & Cord, 2021) shares some concepts with our work in that it promotes network diversity by reducing feature correlations. They use additional networks to estimate mutual information and adversarial training, incurring extra training costs.

Spurious correlations of features. It is generally known that trained neural networks frequently rely on features that are predictive of the target in the training data, but irrelevant to the underlying true labeling function (e.g., backgrounds co-occurring with foreground objects) (Geirhos et al., 2020). Because depending solely on these spurious features impairs generalization performance, several methods are proposed to avoid them (Arjovsky et al., 2019; Kirichenko et al., 2022; Pagliardini et al., 2022). For example, the distributionally robust optimization (DRO) framework (Sagawa et al., 2020) optimizes the worst-case loss over a set of pre-specified groups in the training data instead of the averaged loss to obtain invariant features across groups. In this light, our ensemble method addresses a similar issue by gathering plausible (both spurious and non-spurious) features that explain training data and improves generalization performance even in the presence of distributional corruptions. Note that these studies generally assume that input data from possible test distributions (or out of distributions) are accessible in training, which we do not assume in this study.

6. Conclusion

We have introduced a feature-space particle inference framework for neural network ensembles, which encourages each ensemble member to exploit various data views. Our extensive experiments have shown that feature-space inference significantly outperforms traditional Deep Ensembles and weight/function-space inference in terms of accuracy, uncertainty estimation, and robustness.

Although we consider only a final linear layer as a shared classifier in this work, extending shared parts to shallower layers may be engaging in the future. This may suggest which level of feature variation contributes to ensemble performance. For more efficient implementations, we can use the rank-1 parametrization of an ensemble (Wen et al., 2020; Dusenberry et al., 2020) as a backbone feature extractor. Reducing the evaluation of the repulsive term is also promising, especially for distributed training. Furthermore, we can readily apply the recently developed techniques of particle inference (Gallego & Insua, 2018; Zhang et al., 2020a) for further performance gains.

Acknowledgements

We would like to thank Mitsuru Ambai and Yuichi Yoshida for helpful feedback on the paper.

References

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, 2020.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

Chen, C., Zhang, R., Wang, W., Li, B., and Chen, L. A unified particle-optimization framework for scalable Bayesian sampling. In *Uncertainty in Artificial Intelligence*, pp. 746–755, 2018.

Chen, D., Mei, J.-P., Wang, C., Feng, Y., and Chen, C. Online knowledge distillation with diverse peers. In *AAAI Conference on Artificial Intelligence*, pp. 3430–3437, 2020.

Chen, P. and Ghattas, O. Projected Stein variational gradient descent. In *Neural Information Processing Systems*, pp. 1947–1958, 2020.

D'Angelo, F. and Fortuin, V. Repulsive deep ensembles are Bayesian. In *Neural Information Processing Systems*, 2021.

D'Angelo, F., Fortuin, V., and Wenzel, F. On Stein variational neural network ensembles. *arXiv preprint arXiv:2106.10760*, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Dusenberry, M., Jerfel, G., Wen, Y., Ma, Y., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. Efficient and scalable Bayesian neural nets with rank-1 factors. In *International Conference on Machine Learning*, pp. 2782–2792, 2020.

Dvornik, N., Schmid, C., and Mairal, J. Diversity with cooperation: Ensemble methods for few-shot classification. In *International Conference on Computer Vision*, pp. 3723–3731, 2019.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

Favaro, S., Peluchetti, S., and Fortini, S. Stable behaviour of infinitely wide deep neural networks. In *Artificial Intelligence and Statistics*, pp. 1137–1146, 2020.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Gallego, V. and Insua, D. R. Stochastic gradient MCMC with repulsive forces. *arXiv preprint arXiv:1812.00071*, 2018.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Gelman, A. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian Analysis*, 1(3):515–534, 2006.

Gelman, A. and Yao, Y. Holes in Bayesian statistics. *Journal of Physics G: Nuclear and Particle Physics*, 48(1): 014002, 2020.

Graves, A. Practical variational inference for neural networks. In *Neural Information Processing Systems*, pp. 2348–2356, 2011.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017.

Gustafsson, F. K., Danelljan, M., and Schon, T. B. Evaluating scalable Bayesian deep learning methods for robust computer vision. In *Computer Vision and Pattern Recognition Workshops*, pp. 318–319, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

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

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for Bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pp. 1169–1179, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Neural Information Processing Systems*, pp. 2575–2583, 2015.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, 2009.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neural Information Processing Systems*, pp. 6405–6416, 2017.

Li, Y. and Turner, R. E. Gradient estimators for implicit models. In *International Conference on Learning Representations*, 2018.

Liu, C., Zhuo, J., Cheng, P., Zhang, R., and Zhu, J. Understanding and accelerating particle-based variational inference. In *International Conference on Machine Learning*, pp. 4082–4092, 2019.

Liu, Q. Stein variational gradient descent as gradient flow. In *Neural Information Processing Systems*, pp. 3115–3123, 2017.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Neural Information Processing Systems*, pp. 2378–2386, 2016.

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

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for Bayesian uncertainty in deep learning. In *Neural Information Processing Systems*, pp. 13153–13164, 2019.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

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

Neal, R. M. *Bayesian learning for neural networks*. Springer Science & Business Media, 1996.

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *International Conference on Learning Representations Workshops*, 2015.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Neural Information Processing Systems*, pp. 13991–14002, 2019.

Pagliardini, M., Jaggi, M., Fleuret, F., and Karimireddy, S. P. Agree to disagree: Diversity through disagreement for better transferability. *arXiv preprint arXiv:2202.04414*, 2022.

Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pp. 4970–4979, 2019.

Rame, A. and Cord, M. DICE: Diversity in deep ensembles via conditional redundancy adversarial estimation. In *International Conference on Learning Representations*, pp. 1900–1908, 2021.

Rasmussen, C. E., Williams, C. K., and Bach, F. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Ritter, H., Botev, A., and Barber, D. A scalable Laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020.

Schölkopf, B., Smola, A. J., and Bach, F. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

Scott, D. W. Feasibility of multivariate density estimates. *Biometrika*, 78(1):197–205, 1991.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision*, pp. 618–626, 2017.

Shi, J., Sun, S., and Zhu, J. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pp. 4644–4653, 2018.

Tran, B.-H., Rossi, S., Milios, D., and Filippone, M. All you need is a good functional prior for Bayesian deep learning. *arXiv preprint arXiv:2011.12829*, 2020.

Tsybakov, A. B. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166, 2004.

Villani, C. *Optimal transport: old and new*, volume 338. Springer, 2009.

Wang, Y., Chen, P., and Li, W. Projected Wasserstein gradient descent for high-dimensional Bayesian inference. *arXiv preprint arXiv:2102.06350*, 2021.

Wang, Z., Ren, T., Zhu, J., and Zhang, B. Function space particle optimization for Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, pp. 681–688, 2011.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *International Conference on Learning Representations*, 2018.

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

Wenzel, F., Roth, K., Veeling, B., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the Bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, pp. 10248–10259, 2020a.

Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. Hyperparameter ensembles for robustness and uncertainty quantification. In *Neural Information Processing Systems*, pp. 6514–6527, 2020b.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In *Neural Information Processing Systems*, pp. 4151–4161, 2017.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. In *Neural Information Processing Systems*, pp. 4697–4708, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference*, pp. 87.1–87.12, 2016.

Zhang, J., Zhang, R., Carin, L., and Chen, C. Stochastic particle-optimization sampling and the non-asymptotic convergence theory. In *Artificial Intelligence and Statistics*, pp. 1877–1887, 2020a.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *International Conference on Learning Representations*, 2020b.

Appendix

A. Algorithm

Here we outline the detailed procedure of feature-WGD, possibly under model parallel execution, in Algorithm 1. Although feature-WGD requires additional communication cost (`MPI_Allgather`) to calculate interactions between ensemble members, communicated variables are not so high-dimensional (at most {feature dimension} × {batch size}) compared to the gradient communication in standard data parallel executions, which requires parameter dimension communications. Empirically, these overheads are negligible in practical ensemble size (~ 10).

B. Implementation Details

B.1. Training schemes

Classical hyperparameters in the literature are taken from (Chen et al., 2020). We use stochastic gradient descent with Nesterov momentum for optimization, and adopt standard augmentation schemes (random crop and horizontal flip). Note that these hyperparameter are common to all methods including Deep Ensembles and {weight, function, feature}-WGD.

Table 7. Hyperparameter values for training on CIFAR-10, CIFAR-100, and ImageNet.

DATASET	CIFAR-10	CIFAR-100	IMAGENET
EPOCH	300	300	90
BATCH SIZE	128	128	256
BASE LEARNING RATE	0.1	0.1	0.1
LR DECAY RATIO	0.1	0.1	0.1
LR DECAY EPOCHS	$[150, 225]$	$[150, 225, 250]$	$[30, 60]$
MOMEMTUM	0.9	0.9	0.9
WEIGHT DECAY	5×10^{-4}	5×10^{-4}	1×10^{-4}

B.2. Prior parameters

Because feature values after global average pooling are positive in ResNets, we consider priors (normal, Cauchy, and uniform) to be supported only on the positive parts as follows:

$$\mathrm{HalfNormal}(x; \sigma) = \begin{cases} \sqrt{\frac{2}{\pi\sigma^2}} \mathrm{e}^{-x^2/2\sigma^2} & x \geq 0, \\ 0 & x < 0 \end{cases}$$

$$\mathrm{HalfCauchy}(x; \sigma) = \begin{cases} \frac{2}{\pi\sigma} \frac{1}{1+x^2/\sigma^2} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

In Table 8-12, we present prior parameters on each experiment in Section 4.

Table 8. Prior parameters for Wide ResNet-16-4 on CIFAR-10 in Table 1.

METHOD	WEIGHT-WGD	FUNCTION-WGD	FEATURE-WGD
PRIOR	NORMAL	CAUCHY	CAUCHY
PRIOR SCALE $1/\sigma^2$	1×10^{-3}	1×10^{-6}	1×10^{-3}
PROJECTION DIM r	5	5	5

C. Additional Experiments

Effect of ensembling. To separate the regularization effect induced by the prior on features from the improvement of feature-WGD, we compare a single model trained with the feature prior term (feature-MAP) to one trained without it

Algorithm 1 feature-WGD (in parallel)

```
Input: training data $\mathcal{D}$, the number of optimization steps $T$, step size $\{\alpha_t\}_{t=1}^{T}$, weight decay parameter $\lambda$, projection dimension $r$
Output: optimized parameters $\{w_i\}_{i=1}^{n}, \theta$.
Initialize parameters $\{w_i\}_{i=1}^{n}, \theta$.
for $t = 1$ to $T$ do
    Draw a mini-batch $\{x_b, y_b\}_{b=1}^{B} \sim \mathcal{D}$.
    for $i = 1$ to $n$ do
        Construct a feature vector through feed-forwarding:
            $\mathbf{h}_i = \mathrm{vec}(\{h(x_b; w)\}_b).$
        Calculate gradients:
            $\mathbf{g}_i^{\mathrm{data}} = \nabla_{\mathbf{h}_i} \log p(\{y_b\}_b | \mathbf{h}_i),$
            $\mathbf{g}_i^{\mathrm{prior}} = \nabla_{\mathbf{h}_i} \log p(\mathbf{h}_i).$
    end for
    (Perform MPI_Allgather for $\left\{\mathbf{g}_i^{\mathrm{data}}\right\}_{i=1}^{n}$ .)
    Construct basis:
        $\Psi, \Sigma, V = \mathrm{SVD}([\mathbf{g}_1^{\mathrm{data}}, \mathbf{g}_2^{\mathrm{data}}, \ldots, \mathbf{g}_{\mathrm{n}}^{\mathrm{data}}]),$
        $\Psi_r = \Psi[:, :r].$
    for $i = 1$ to $n$ do
        Project a feature vector:
            $\mathbf{z}_i = \Psi_r^{\top} \mathbf{h}_i.$
    end for
    (Perform MPI_Allgather for $\{\mathbf{z}_i\}_{i=1}^{n}$ .)
    for $i = 1$ to $n$ do
        Calculate a feature update direction:
            $v_i^{\mathbf{h}} = \mathbf{g}_i^{\mathrm{data}} + \mathbf{g}_i^{\mathrm{prior}} - \Psi_r \frac{\sum_{j=1}^{n} \nabla_{\mathbf{z}_i} k(\mathbf{z}_i, \mathbf{z}_j)}{\sum_{j=1}^{n} k(\mathbf{z}_i, \mathbf{z}_j)}.$
        Calculate a parameter update direction through back-propagation:
            $v_i^{w} = \frac{1}{B} \left( \frac{\partial \mathbf{h}_i}{\partial w_i} \right)^{\top} v_i^{\mathbf{h}} - \lambda w_i,$
            $v_i^{\theta} = \frac{1}{B} \sum_{b=1}^{B} \nabla_{\theta} \log p\left(y_b | c(h(x_b; w_i); \theta)\right) - \lambda \theta.$
        Update feature extractor parameters:
            $w_i \leftarrow w_i + \alpha_t v_i^{w},$
    end for
    (Perform MPI_Allgather for $\left\{v_i^{\theta}\right\}_{i=1}^{n}$ .)
    Update classifier parameters:
        $\theta \leftarrow \theta + \frac{\alpha_t}{n} \sum_{i=1}^{n} v_i^{\theta}.$
end for
```

Table 9. Prior parameters for Wide ResNet-16-4 on CIFAR-100 in Table 2.

METHOD	WEIGHT-WGD	FUNCTION-WGD	FEATURE-WGD
PRIOR	NORMAL	CAUCHY	CAUCHY
PRIOR SCALE $1/\sigma^2$	1×10^{-3}	1×10^{-6}	5×10^{-3}
PROJECTION DIM r	5	5	5

Table 10. Prior parameters for ResNet-50 on ImageNet in Table 3.

METHOD	FEATURE-WGD
PRIOR	CAUCHY
PRIOR SCALE $1/\sigma^2$	2×10^{-3}
PROJECTION DIM r	5

Table 11. Prior parameters for ResNet-32 and Wide ResNet-28-2 on CIFAR-100 in Table 4.

ARCHITECTURE	RESNET-32×4	WRN-28-2×3
PRIOR	CAUCHY	CAUCHY
PRIOR SCALE $1/\sigma^2$	1×10^{-3}	1×10^{-3}
PROJECTION DIM r	4	3

Table 12. Prior parameters for Wide ResNet-16-4 on CIFAR-100 with various priors in Table 5.

PRIOR	NORMAL	CAUCHY	UNIFORM
PRIOR SCALE $1/\sigma^2$	5×10^{-3}	5×10^{-3}	–
PROJECTION DIM r	5	5	5

(Single). The results summarized in Table 13 show that imposing prior itself does not improve calibration scores on a single model, whereas accuracy is slightly improved. From this, we can see that the superior performance of feature-WGD to Deep Ensembles comes not from the regularization scheme on each model but ensembling and interaction between models.

Table 13. Results for single Wide ResNet-16-4 on CIFAR-100, trained with (feature-MAP) and without (Single) the feature prior term. Averaged over 5 seeds.

METHOD	ACCURACY	NLL	BRIER	ECE
SINGLE	77.4	0.835	0.316	0.030
FEATURE-MAP	77.6	0.883	0.318	0.044

Ablation study on the projection dimension parameter r. Here we show the influence of the projection dimension parameter r on CIFAR-100 setting in Section 4.1 in Table 14.

Table 14. Ablation study on the projection dimension r for Wide ResNet-16-4 on CIFAR-100. Averaged over 3 seeds.

	$r = 3$	$r = 5$	$r = 10$	NO PROJECTION
ACCURACY	82.7	82.9	82.7	82.7

D. Additional figures

We put results for CIFAR-10-C (Figure 5) and CIFAR-100-C (Figure 6) in Section 4.1 across corruption types for each corruption severity.

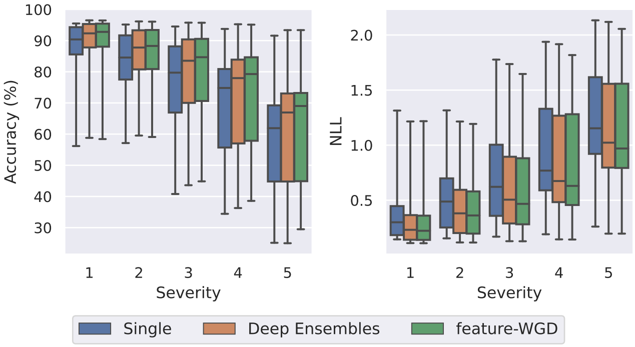

Figure 5. Results for Wide ResNet-16-4 on CIFAR-10-C with an ensemble size of 5. We plot accuracy and NLL evaluated on 15 corruption types for varying corruption severity 1-5.

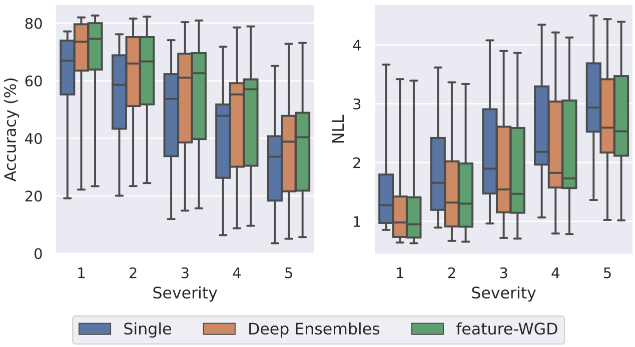

Figure 6. Results for Wide ResNet-16-4 on CIFAR-100-C with an ensemble size of 5. We plot accuracy and NLL evaluated on 15 corruption types for varying corruption severity 1-5.