Repulsive Deep Ensembles are Bayesian

Francesco D'Angelo [1] **Vincent Fortuin** [1]

Abstract

Deep ensembles have recently gained popularity in the deep learning community for their conceptual simplicity and efficiency. However, maintaining functional diversity between ensemble members that are independently trained with gradient descent is challenging. This issue does not only affect the quality of its predictions, but even more so the uncertainty estimates of the ensemble, and thus its performance on out-of-distribution data. We hypothesize that this limitation can be overcome by discouraging different ensemble members from collapsing to the same function. To this end, we introduce a kernelized repulsive term in the update rule of the deep ensembles. We show that this simple modification not only enforces and maintains diversity among the members but, even more importantly, the training dynamics of our proposed repulsive ensembles follow the Wasserstein gradient flow of the KL divergence with the true posterior. We study repulsive terms in weight and function space and empirically compare their performance to standard ensembles and Bayesian baselines on synthetic and real-world prediction tasks.

1. Introduction

There have been many recent advances on the theoretical properties of sampling algorithms for approximate Bayesian inference. Particularly worth mentioning is the work of Jordan et al. (1998), who reinterpret Markov Chain Monte Carlo (MCMC) as a gradient flow of the KL divergence over the Wasserstein space. Following this direction, Liu & Wang (2016) recently proposed the Stein Variational Gradient Descent (SVGD) method to perform approximate Wasserstein gradient descent. Conceptually, this method, that belongs to the family of particle-optimization variational inference (POVI), introduces a repulsive force through a kernel to evolve a set of samples towards high-density regions of the target distribution without collapsing. However, when dealing with neural networks, the Bayesian approach is not the only one capable of offering calibrated models that allow for uncertainty estimation; ensembles of neural networks (so-called *deep ensembles*) have indeed achieved great success recently, both in terms of predictive performance (Lakshminarayanan et al., 2017; Wenzel et al., 2020b) as well as uncertainty estimation (Ovadia et al., 2019), and constitute the main competitor to the Bayesian counterpart. That being said, while they might allow for the averaging of predictions over several hypotheses, they do not offer any guarantees for the diversity between those hypotheses nor do they benefit from the advantages of the probabilistic Bayesian framework. In this work, we show how the introduction of a repulsive term between the members in the ensemble, inspired by SVGD, not only naïvely guarantees the diversity among the members, avoiding their collapse in the parameter space, but also allows for a reformulation of the method as a gradient flow of the KL divergence in the Wasserstein space of distributions. It thus allows to reformulate deep ensembles with repulsion as a truly Bayesian method. Nevertheless, BNN inference in weight space can lead to degenerate solutions, due to the overparametrization of these models. That is, several samples could have very different weights but map to the same function, thus giving a false sense of diversity in the ensemble. This property, that we will refer to as *non-identifiability* of neural networks (see Appendix B), can lead to redundancies in the ensemble members. To overcome this issue, Wang et al. (2019) recently introduced a new method to extend POVI methods to function space. Here, we also study an update rule that allows for an approximation of the gradient flow of the KL divergence in function space instead of weight space. We make the following contributions:

- We introduce a kernelized repulsion to the gradient updates of deep ensembles that endows them with Bayesian properties.
- We show that a specific repulsion approximates Wasserstein gradient flows of the KL divergence and can be used both in weight and function space.
- We compare these proposed methods empirically and theoretically to standard deep ensembles and SVGD and highlight their different guarantees.

[1] ETH Zürich, Zürich, Switzerland. Correspondence to: Francesco D'Angelo <fdangelo@student.ethz.ch>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.,

2. Repulsive Deep Ensembles

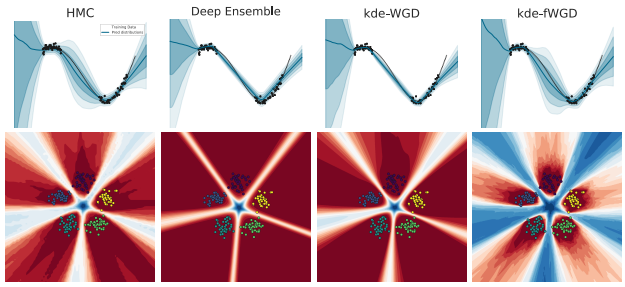

Figure 1. **1d regression(top)**: all methods in the weight space are unable to capture the epistemic uncertainty between the two clusters of training data points. Conversely, the Wasserstein ensemble in function space maintain this uncertainty and leads to a posterior predictive that very closely resembles the one obtained with the HMC sampling. **2d classification (bottom):** similarly, we can observe that the weight-space methods are overconfident and do not capture the uncertainty well. Conversely, our fWGD method is confident (low entropy) around the data but not out-of-distribution, leading to a *distance-aware* uncertainty estimation, a property that translates into confident predictions only in the proximity of the training data, allowing for a principled OOD detection.

While approximating the posterior of deep neural networks (or sampling from it) is a challenging task, performing maximum a posteriori (MAP) estimation, which corresponds to finding the mode of the posterior, is usually simple. Ensembles of neural networks use the non-convexity of the MAP optimization problem to create a collection of K independent—and possibly different— solutions. Considering n weight configurations of neural networks $\{\mathbf{w}_i\}_{i=1}^n$ with $\mathbf{w}_i \in \mathbb{R}^d$, the dynamics of the ensemble under the gradient of the posterior lead to an update rule at iteration t:

$$\begin{aligned} \mathbf{w}_i^{t+1} &\leftarrow \mathbf{w}_i^t + \epsilon_t \phi(\mathbf{w}_i^t) \\ \text{with} \qquad \phi(\mathbf{w}_i^t) &= \nabla_{\mathbf{w}_i^t} \log p(\mathbf{w}_i^t|\mathcal{D}) \,, \end{aligned} \tag{1}$$

with step size ϵ_t. Ensemble methods have a long history (e.g., Levin et al., 1990; Hansen & Salamon, 1990; Breiman, 1996) and were recently revisited for neural networks (Lakshminarayanan et al., 2017) and coined *deep ensembles*. The predictions of the different members are combined to create a predictive distribution by using the solutions to compute the Bayesian model average (BMA) in equation 9. Recent works (Ovadia et al., 2019) have shown that deep ensembles can outperform some of the Bayesian approaches for uncertainty estimation. Even more recently, Wilson & Izmailov (2020) argued that deep ensembles can be considered as an approach to Bayesian model averaging. Despite these ideas, the ability of deep ensembles to efficiently average over multiple hypotheses and to explore the functional landscape of the posterior distribution studied in (Fort et al., 2019) does not make the method necessarily Bayesian.

From a practical standpoint, many methods were recently proposed to improve this diversity without compromising the individual accuracy (Wenzel et al., 2020b; Huang et al., 2017b; Rame & Cord, 2021). However, the absence of a constraint that prevents particles from converging to the same mode limits the possibility of improvement by introducing more ensemble members. This means that any guarantees to converge to different modes must exclusively rely on: 1. The randomness of the initialization; 2. The noise in the estimation of the gradients due to minibatching; 3. The number of local optima that might be reached during gradient descent. Moreover, the recent study of Geiger et al. (2020) showed how the empirical test error of the ensemble converges to the one of a single trained model when the number of parameters goes to infinity, leading to deterioration of the performance. In other words, the bigger the model, the harder it is to maintain diversity in the ensemble and avoid collapse to the same solution.

Repulsive force in weight space To overcome the aforementioned limitations of standard deep ensembles, we introduce, inspired by SVGD (Liu & Wang, 2016), a deep ensemble with members that interact with each other through a repulsive component. Using a kernel function to model this interaction, the single models repel each other based on their position in the weight space, so that two members can never assume the same weights. Interactions based on weight sharing have also been studied (Oswald et al., 2021) but they do not actively enforce diversity in the ensemble. Considering a stationary kernel $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ acting in the parameter space of the neural networks, a repulsive term can be parameterized through its gradient:

$$\phi(\mathbf{w}_i^t) = \nabla_{\mathbf{w}_i^t} \log p(\mathbf{w}_i^t|\mathcal{D}) - \beta\left(\{\mathbf{w}_j^t\}_1^n\right) \sum_{j=1}^n \nabla_{\mathbf{w}_i^t} k(\mathbf{w}_i^t, \mathbf{w}_j^t) \,, \tag{2}$$

with $\beta\left(\{\mathbf{w}_j^t\}_1^n\right) > 0$. To get an intuition for the behavior of this repulsive term and its gradients, we can consider the RBF kernel $k(\mathbf{w}, \mathbf{w}') = \exp\left(-\frac{1}{h}||\mathbf{w} - \mathbf{w}'||^2\right)$ with lengthscale h and notice how its gradient $\nabla_{\mathbf{w}_i^t} k(\mathbf{w}_i^t, \mathbf{w}_j^t) = \frac{2}{h}(\mathbf{w}_j^t - \mathbf{w}_i^t) k(\mathbf{w}_i^t, \mathbf{w}_j^t)$ drives $\mathbf{w}_i$ away from its neighboring members $\mathbf{w}_j$, thus creating a repulsive effect.

Repulsive force in function space To overcome the aforementioned overparameterization issue, the update in equation 2 can be formulated in function space instead of weight space. Let $\boldsymbol{f} : \mathbf{w} \mapsto f(\cdot\,; \mathbf{w})$ be the map that maps a configuration of weights $\mathbf{w} \in \mathbb{R}^d$ to the corresponding neural network regression function and denote $\boldsymbol{f}_i := f(\cdot\,; \mathbf{w}_i)$ the function with a certain configuration of weights $\mathbf{w}_i$. We can now consider n particles in function space $\{\boldsymbol{f}_i\}_{i=1}^n$ with $\boldsymbol{f} \in \mathcal{F}$ and model their interactions with a general positive definite kernel $k(\cdot,\cdot)$. We also consider the implicit functional likelihood $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{f})$, determined by the measure $p(\boldsymbol{y}|\boldsymbol{x}, \mathbf{w})$ in the weight space, as well as the functional

prior $p(\boldsymbol{f})$, which can either be defined separately (e.g., using a GP) or modeled as a push-forward measure of the weight-space prior $p(\mathbf{w})$. Together, they determine the posterior in function space $p(\boldsymbol{f}|\mathcal{D})$. The functional evolution of a particle can then be written as $\boldsymbol{f}_i^{t+1} \leftarrow \boldsymbol{f}_i^t + \epsilon_t \phi(\boldsymbol{f}_i^t)$,

$$\phi(\boldsymbol{f}_i^t) = \nabla_{\boldsymbol{f}_i^t} \log p(\boldsymbol{f}_i^t|\mathcal{D}) - \alpha \left(\{\boldsymbol{f}_j^t\}_{j=1}^n\right) \sum_{j=1}^n \nabla_{\boldsymbol{f}_i^t} k(\boldsymbol{f}_i^t, \boldsymbol{f}_j^t) \,. \tag{3}$$

However, computing the update in function space is neither tractable nor practical, which is why two additional considerations are needed. The first one regards the infinite dimensionality of function space, which we circumvent using a canonical projection into a subspace:

Definition 1 (Canonical projection)**.** *For any $A \subset \mathcal{X}$, we define $\pi_A : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}^A$ as the canonical projection onto A i.e. $\pi_A(f) = \{f(a)\}_{a \in A}$.*

In other words, the kernel will not be evaluated directly in function space, but on the projection $k\big(\pi_B(f), \pi_B(f')\big)$, with B being a subset of the input space given for instance by a batch of training datapoints. The second consideration is to project this update back into the parameter space and evolve a set of particles there, because ultimately we are interested in representing the functions by parametrized neural networks. For this purpose, we can use the Jacobian of the i-th particle as a projector:

$$\phi(\mathbf{w}_i^t) = \left(\frac{\partial \boldsymbol{f}_i^t}{\partial \mathbf{w}_i^t}\right)^\top \Bigg[\nabla_{\boldsymbol{f}_i^t} \log p(\boldsymbol{f}_i^t|\mathcal{D}) - \alpha \left(\left\{\pi_B(\boldsymbol{f}_j^t)\right\}_{j=1}^n\right) \sum_{j=1}^n \nabla_{\boldsymbol{f}_i^t} k\left(\pi_B(\boldsymbol{f}_i^t), \pi_B(\boldsymbol{f}_j^t)\right)\Bigg] \,. \tag{4}$$

Repulsive deep ensembles are Bayesian Thus far, we represented the repulsive force as a general function of the gradients of a kernel. In this section, we show how to determine the explicit form of the repulsive term, such that the resulting update rule is equivalent to the discretization of the gradient flow dynamics of the KL divergence in Wasserstein space. We begin by introducing the concepts of particle approximation and gradient flow.

Particle approximation A particle-based approximation of a target measure depends on a set of weighted samples $\{(x_i, w_i)\}_{i=1}^n$, for which an empirical measure can be defined as:

$$\rho(x) = \sum_{i=1}^n w_i \, \delta(x - x_i) \,, \tag{5}$$

where $\delta(\cdot)$ is the Dirac delta function and the weights w_i satisfy $w_i \in [0, 1]$ and $\sum_{i=1}^n w_i = 1$. To approximate a target distribution $\pi(x)$ using the empirical measure, the particles and their weights need to be selected in a principled manner that minimizes some measure of distance between $\pi(x)$ and $\rho(x)$ (e.g., a set of N samples with weights $w_i = 1/N$ obtained using an MCMC method).

Gradient flow in parameter space Given a smooth function $J : \mathbb{R}^d \to \mathbb{R}$ in Euclidean space, we can minimize it by creating a path that follows its negative gradient starting from some initial conditions x_0. The curve $x(t)$ with starting point x_0 described by that path is called *gradient flow*. The dynamics and evolution in time of a considered point in the space under this minimization problem can be described as the following ODE: $\frac{dx}{dt} = -\nabla J(x)$.

We can now extend this concept to the space of probability distributions (*Wasserstein gradient flow*) (Ambrosio et al., 2008). Let us consider the space of probability measures $\mathcal{P}_2(\mathcal{M})$, that is the set of probability measures with finite second moments defined on the manifold $\mathcal{M}$. Taking $\Pi(\mu, \nu)$ as the set of joint probability measures with marginals μ, ν, we can define the Wasserstein metric on $\mathcal{P}_2(\mathcal{M})$ as: $W_2^2(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int |x - y|^2 \, \mathrm{d}\pi(x, y)$. Considering the optimization problem of a functional $J : \mathcal{P}_2(\mathcal{M}) \to \mathbb{R}$, such as the KL divergence between the particle approximation in equation 5 and the posterior $\pi(x)$,

$$\inf_{\rho \in \mathcal{P}_2(\mathcal{M})} D_{KL}(\rho, \pi) = \int_{\mathcal{M}} (\log \rho(x) - \log \pi(x)) \rho(x) \, \mathrm{d}x \,,$$

the evolution in time of the measure ρ under the equivalent of the gradient, the Wasserstein gradient flow, is described by the *Liouville equation*:

$$\begin{aligned} \frac{\partial \rho(x)}{\partial t} &= \nabla \cdot \left(\rho(x) \nabla \frac{\delta}{\delta \rho} D_{KL}(\rho, \pi)\right) \\ &= \nabla \cdot \left(\rho(x) \nabla \big(\log \rho(x) - \log \pi(x)\big)\right) \,, \end{aligned} \tag{6}$$

where $\nabla \frac{\delta}{\delta \rho} D_{KL}(\rho, \pi) =: \nabla_{\mathcal{W}_2} D_{KL}(\rho, \pi)$ is the Wasserstein gradient and the operator $\frac{\delta}{\delta \rho} : \mathcal{P}_2(\mathcal{M}) \to \mathbb{R}$ represents the functional derivative or first variation (see Appendix D for more details). In the particular case of the KL functional, we can recover the Fokker-Planck equation, $\frac{\partial \rho(x)}{\partial t} = \nabla \cdot \big(\rho(x) \nabla (\log \rho(x) - \log \pi(x))\big) = -\nabla \cdot \big(\rho(x) \nabla \log \pi(x)\big) + \nabla^2 \rho(x)$, that admits as unique stationary distribution $\pi(x)$. The deterministic particle dynamics ODE (Ambrosio & Crippa, 2014) related to equation 6, namely mean-field Wasserstein dynamics, is then given by: $\frac{dx}{dt} = -\nabla \big(\log \rho(x) - \log \pi(x)\big)$. Considering a discretization of the previous equation for a particle system $\{x\}_{i=1}^n$ and small stepsize ϵ_t, we can rewrite it as:

$$x_i^{t+1} = x_i^t + \epsilon_t \big(\nabla \log \pi(x_i^t) - \nabla \log \rho(x_i^t)\big) \,. \tag{7}$$

Unfortunately, we do not have access to the analytical form of the gradient $\nabla \log \rho$, so an approximation is needed. At

	FashionMNIST						CIFAR10					
	AUROC(H)	**AUROC(MD)**	**Accuracy**	$\mathbf{MD_o/MD_t}$	**ECE**	**NLL**	**AUROC(H)**	**AUROC(MD)**	**Accuracy**	$\mathbf{MD_o/MD_t}$	**ECE**	**NLL**
Deep ens.	0.958±0.001	0.975±0.001	91.122±0.013	6.394±0.001	**0.012±0.001**	0.116±0.001	**0.843±0.004**	0.736±0.005	85.552±0.076	1.667±0.008	0.049±0.001	0.277±0.001
SVGD	0.960±0.001	0.973±0.001	91.134±0.024	6.315±0.019	0.014±0.001	0.116±0.001	0.825±0.001	0.710±0.002	85.142±0.017	1.567±0.004	0.052±0.001	0.287±0.001
f-SVGD	0.956±0.001	0.975±0.001	89.884±0.015	5.652±0.009	0.013±0.001	0.150±0.001	0.783±0.001	0.712±0.001	84.510±0.031	1.624±0.003	0.049±0.001	0.292±0.001
kde-WGD	0.960±0.001	0.970±0.001	91.238±0.019	6.587±0.019	0.014±0.001	0.116±0.001	0.838±0.001	0.735±0.004	**85.904±0.030**	1.661±0.008	0.053±0.001	**0.276±0.001**
sge-WGD	0.960±0.001	0.970±0.001	**91.312±0.016**	6.562±0.007	**0.012±0.001**	0.116±0.001	0.837±0.003	0.725±0.004	85.792±0.035	1.634±0.004	0.051±0.001	**0.275±0.001**
ssge-WGD	0.968±0.001	0.979±0.001	91.198±0.024	6.522±0.009	**0.012±0.001**	0.130±0.001	0.832±0.003	0.731±0.005	85.638±0.038	1.655±0.001	0.049±0.001	**0.276±0.001**
kde-fWGD	**0.971±0.001**	**0.980±0.001**	91.260±0.011	6.887±0.015	0.015±0.001	**0.115±0.001**	0.791±0.002	**0.758±0.002**	84.888±0.030	**1.749±0.005**	**0.044±0.001**	0.282±0.001
sge-fWGD	0.969±0.001	0.978±0.001	91.192±0.013	7.076±0.004	0.015±0.001	**0.115±0.001**	0.795±0.001	0.754±0.002	84.766±0.060	1.729±0.002	0.047±0.001	0.288±0.001
ssge-fWGD	**0.971±0.001**	**0.980±0.001**	91.240±0.022	**6.951±0.005**	0.016±0.001	**0.115±0.001**	0.792±0.002	0.752±0.002	84.762±0.034	1.723±0.005	0.046±0.001	0.286±0.001

Table 1. **BNN image classification. AUROC(H)** is the AUROC computed using the entropy, whereas **AUROC(MD)** is computed using the model disagreement. $\mathbf{MD_o/MD_t}$ is the ratio of model disagreement on OOD and test points. The best accuracy is achieved by our WGD methods, while our fWGD methods yield the best OOD detection and functional diversity. All our proposed methods improve over standard deep ensembles in terms of accuracy and diversity, highlighting the effect of our repulsion.

this point, it is crucial to observe the similarity between the discretization of the Wasserstein gradient flow in equation 7 and the repulsive update in equation 2 to notice how, if the kernelized repulsion is an approximation of the gradient of the empirical particle measure, the update rule minimizes the KL divergence between the particle measure and the target posterior. Approximating the density of the particles using a kernel density estimation (KDE, details in App. E) $\tilde{\rho}_t(x) = \sum_{i=1}^n k(x, x_t^i)$, the gradient of its log density is given by (Singh, 1977): $\nabla \log \rho(x_i^t) \approx \frac{\sum_{j=1}^n \nabla_{x_i^t} k(x_i^t, x_j^t)}{\sum_{j=1}^n k(x_i^t, x_j^t)}$. Using this approximation in equation 7 we obtain:

$$x_i^{t+1} = x_i^t + \epsilon_t \left(\nabla \log \pi(x_i^t) - \frac{\sum_{j=1}^n \nabla_{x_i^t} k(x_i^t, x_j^t)}{\sum_{j=1}^n k(x_i^t, x_j^t)} \right), \tag{8}$$

where, if we substitute the posterior for π, we obtain an expression for the repulsive force in equation 2. This gives a consistent formulation to specify β in equation 2 so that $\beta\left(\{\mathbf{w}_j^t\}_{j=1}^n\right) := \sum_{j=1}^n k(\mathbf{w}_i^t, \mathbf{w}_j^t)$ and shows that this regularizer does not only encourage diversity of the ensemble members and thus avoids collapse, but surprisingly—in the asymptotic limit of $n \to \infty$, where the KDE approximation is exact—also converges to the true Bayesian posterior! A similar argument can be derived for the function space case (see Appendix G). We will call the weight-space method *kde-WGD* and the function-space one *kde-fWGD*. Nevertheless, approximating the gradient of the empirical measure with the KDE can lead to suboptimal performance, as already studied by Li & Turner (2017). Hence, different kernel-based approximations have been developed. For instance, the Stein Gradient Estimator (SGE) (Li & Turner, 2017) and Spectral Stein Estimator (SSGE) (Shi et al., 2018) can be used to formulate different repulsive forces that maintain the same properties of the KDE, see App. H for details.

3. Experiments

In this section, we compare our repulsive ensemble with deep ensembles and SVGD on sampling (see App. I), regression, and classification tasks, and real-world image classification. To assess the diversity in function space, we measure the model disagreement (MD) (see App. C).

Synthetic tasks. We first asses the different methods in fitting a BNN posterior on a one-dimensional regression task. The results are reported in Figure 1 (top), consisting of the mean prediction and $\pm 1, 2, 3$ standard deviations of the predictive distribution. Next, we visualize the quality of uncertainty estimation of the methods on a two-dimensional classification setting. The entropy of the predictive posteriors are reported in Figure 1 (bottom).

Image classification. We first train a feed-forward neural network with three hidden layers composed of 100 ReLU units on the FashionMNIST dataset (Xiao et al., 2017) and use the MNIST dataset (LeCun, 1998) as an out-of-distribution (OOD) task. The results are reported in Table 1 (left). Secondly, we use a ResNet32 architecture (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009) with the SVHN dataset (Netzer et al., 2011) as OOD data. The results are in Table 1 (right). We can see that our WGD methods achieve the best accuracy, while our fWGD methods achieve the best diversity and OOD detection. The only exception is the CIFAR10 experiment, in which the deep ensembles achieve the best OOD detection using the entropy (but not using the model disagreement).

4. Conclusion

We have presented a simple and principled way to improve upon standard deep ensemble methods. To this end, we have shown that the introduction of a kernelized repulsion between members of the ensemble not only improves the accuracy of the predictions but—even more importantly—can be seen as Wasserstein gradient descent on the KL divergence, thus transforming the MAP inference of deep ensembles into proper Bayesian inference. Moreover, we have shown that incorporating functional repulsion between ensemble members can improve the quality of the estimated uncertainties on simple synthetic examples and OOD detection on real-world data and can approach the true Bayesian posterior more closely.

References

Ambrosio, L. and Crippa, G. Continuity equations and ode flows with non-smooth velocity. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 144 (6):1191–1244, 2014.

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

Badrinarayanan, V., Mishra, B., and Cipolla, R. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754*, 2015.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Burt, D. R., Ober, S. W., Garriga-Alonso, A., and van der Wilk, M. Understanding variational inference in function-space. *arXiv preprint arXiv:2011.09421*, 2020.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018.

Chen, A. M., Lu, H.-m., and Hecht-Nielsen, R. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

Chen, C., Zhang, R., Wang, W., Li, B., and Chen, L. A unified particle-optimization framework for scalable bayesian sampling. *arXiv preprint arXiv:1805.11659*, 2018.

Chen, J., Wu, X., Liang, Y., Jha, S., et al. Robust out-of-distribution detection in neural networks. *arXiv preprint arXiv:2003.09711*, 2020.

Ciosek, K., Fortuin, V., Tomioka, R., Hofmann, K., and Turner, R. Conservative uncertainty estimation by fitting prior networks. In *International Conference on Learning Representations*, 2019.

D'Angelo, F. and Fortuin, V. Annealed stein variational gradient descent. *arXiv preprint arXiv:2101.09815*, 2021.

Duncan, A., Nuesken, N., and Szpruch, L. On the geometry of stein variational gradient descent. *arXiv preprint arXiv:1912.00894*, 2019.

Dusenberry, M. W., Jerfel, G., Wen, Y., Ma, Y.-a., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. Efficient and scalable bayesian neural nets with rank-1 factors. *arXiv preprint arXiv:2005.07186*, 2020.

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

Fortuin, V. Priors in bayesian deep learning: A review. *arXiv preprint arXiv:2105.06868*, 2021.

Fortuin, V., Garriga-Alonso, A., van der Wilk, M., and Aitchison, L. Bnnpriors: A library for bayesian neural network inference with different prior distributions. *Software Impacts*, pp. 100079, 2021a.

Fortuin, V., Garriga-Alonso, A., Wenzel, F., Rätsch, G., Turner, R., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. *arXiv preprint arXiv:2102.06571*, 2021b.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. *arXiv preprint arXiv:1802.10026*, 2018.

Garriga-Alonso, A. and Fortuin, V. Exact langevin dynamics with stochastic gradients. *arXiv preprint arXiv:2102.01691*, 2021.

Geiger, M., Jacot, A., Spigler, S., Gabriel, F., Sagun, L., d'Ascoli, S., Biroli, G., Hongler, C., and Wyart, M. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.

Hansen, L. K. and Salamon, P. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10): 993–1001, October 1990. ISSN 0162-8828. doi: 10.1109/34.58871. URL https://doi.org/10.1109/34.58871.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. *arXiv preprint arXiv:2007.05864*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hecht-Nielsen, R. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pp. 129–135. Elsevier, 1990.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017a.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. Snapshot ensembles: Train 1, get M for free. *CoRR*, abs/1704.00109, 2017b. URL http://arxiv.org/abs/1704.00109.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Khan, M. E. Scalable marginal likelihood estimation for model selection in deep learning. *arXiv preprint arXiv:2104.04975*, 2021a.

Immer, A., Korzepa, M., and Bauer, M. Improving predictions of bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pp. 703–711. PMLR, 2021b.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8580–8589, 2018.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Korba, A., Salim, A., Arbel, M., Luise, G., and Gretton, A. A non-asymptotic analysis for stein variational gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. 2009.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Levin, E., Tishby, N., and Solla, S. A. A statistical approach to learning and generalization in layered neural networks. *Proc. of the IEEE – Special issue on Neural Networks*, 1990.

Li, Y. and Turner, R. E. Gradient estimators for implicit models. *arXiv preprint arXiv:1705.07107*, 2017.

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. PMLR, 2019.

Liu, Q. Stein variational gradient descent as gradient flow. In *Advances 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 *Advances in neural information processing systems*, pp. 2378–2386, 2016.

Lyle, C., Schut, L., Ru, R., Gal, Y., and van der Wilk, M. A bayesian perspective on training speed and model selection. *Advances in Neural Information Processing Systems*, 33, 2020.

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

Malinin, A., Mlodozeniec, B., and Gales, M. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076*, 2019.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

Matthews, A. G. d. G., Hron, J., Turner, R. E., and Ghahramani, Z. Sample-then-optimize posterior sampling for bayesian linear models. In *NeurIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

Neal, R. M. Bayesian learning for neural networks. 1995.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

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

Oswald, J. V., Kobayashi, S., Sacramento, J., Meulemans, A., Henning, C., and Grewe, B. F. Neural networks with late-phase weights. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=C0qJUx5dxFb.

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 *Advances in Neural Information Processing Systems*, pp. 13991–14002, 2019.

Parzen, E. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3): 1065–1076, 1962.

Rame, A. and Cord, M. Dice: Diversity in deep ensembles via conditional redundancy adversarial estimation. *arXiv preprint arXiv:2101.05544*, 2021.

Roeder, G., Metz, L., and Kingma, D. P. On linear identifiability of learned representations. *arXiv preprint arXiv:2007.00810*, 2020.

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. PMLR, 2018.

Singh, R. S. Improvement on some known nonparametric uniformly consistent estimators of derivatives of a density. *The Annals of Statistics*, pp. 394–399, 1977.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Swiatkowski, J., Roth, K., Veeling, B. S., Tran, L., Dillon, J. V., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. The k-tied normal distribution: A compact parameterization of gaussian mean field posteriors in bayesian neural networks. *arXiv preprint arXiv:2002.02655*, 2020.

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. *arXiv preprint arXiv:1902.09754*, 2019.

Wen, Y., Tran, D., and Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, 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? *arXiv preprint arXiv:2002.02405*, 2020a.

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

Williams, C. K. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

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

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

A. Bayesian Neural Networks

In supervised deep learning, we typically consider a likelihood function $p(\boldsymbol{y}|f(\boldsymbol{x};\mathbf{w}))$ (e.g., Gaussian for regression or Categorical for classification) parameterized by a neural network $f(\boldsymbol{x};\boldsymbol{w})$ and training data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$. In Bayesian neural networks (BNNs), we are interested in the posterior distribution of all likely networks given by $p(\mathbf{w}|\mathcal{D}) \propto \prod_{i=1}^n p(\boldsymbol{y}_i|f(\boldsymbol{x}_i;\mathbf{w}))\, p(\mathbf{w})$, where $p(\mathbf{w})$ is the prior distribution over weights. Crucially, when making a prediction on a test point $\boldsymbol{x}^*$, in the Bayesian approach we do not only use a single parameter $\hat{\mathbf{w}}$ to predict $\boldsymbol{y}^* = f(\boldsymbol{x}^*;\hat{\mathbf{w}})$, but we marginalize over the whole posterior, thus taking all possible explanations of the data into account:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|f(\boldsymbol{x}^*;\mathbf{w}))\, p(\mathbf{w}|\mathcal{D})\, \mathrm{d}\,\mathbf{w} \tag{9}$$

B. Non-identifiable neural networks

Deep neural networks are parametric models able to learn complex non-linear functions from few training instances and thus can be deployed to solve many tasks. Their overparameterized architecture, characterized by a number of parameters much larger than that of training data points, enables them to retain entire datasets even with random labels (Zhang et al., 2016). Even more, this overparameterized regime makes neural network approximations of a given function not unique in the sense that multiple configurations of weights might lead to the same function. Indeed, the output of a feed forward neural network given some fixed input remains unchanged under a set of transformations. For instance, certain weight permutations and sign flips in MLPs leave the output unchanged (Chen et al., 1993). The invariance of predictions and therefore of parameterized functions under a given weight transformation translates to invariance of the likelihood function. This effect is commonly denoted as *non-identifiability* of neural networks (Roeder et al., 2020). More in detail, let $g : (\boldsymbol{x}, \mathbf{w}) \mapsto f(\boldsymbol{x};\mathbf{w})$ be the map that maps a data point $\boldsymbol{x} \in \mathcal{X}$ and a weight vector $\mathbf{w} \in \mathbb{R}^d$ to the corresponding neural network output and denote $\boldsymbol{f}_i := f(\boldsymbol{x};\mathbf{w}_i)$ the output with a certain configuration of weights $\mathbf{w}_i$. Then for any non identifiable pair $\{\mathbf{w}_i, \mathbf{w}_j\} \in \mathcal{W} \subseteq \mathbb{R}^d$ and $\boldsymbol{f}_i, \boldsymbol{f}_j \in \mathcal{F}$ their respective functions:

$$\boldsymbol{f}_i = \boldsymbol{f}_j \implies p(\mathbf{w}_i|\mathcal{D}) = p(\mathbf{w}_j|\mathcal{D}) \not\implies \mathbf{w}_i = \mathbf{w}_j\,.$$

Strictly speaking, the map $g : \mathcal{X} \times \mathcal{W} \to \mathcal{F}$ is not injective (many to one). Denoting by T the class of transformations under which a neural network is non-identifiable in the weights space, it is always possible to identify a cone $K \subset \mathbb{R}^d$ such that for any parameter configuration $\mathbf{w}$ there exist a point $\eta \in K$ and a transformation $\tau \in T$ for which it holds that $\tau(\eta) = \mathbf{w}$. This means that every parameter configuration has an equivalent on the subset given by the cone (Hecht-Nielsen, 1990). Modern neural networks containing convolutional and max-pooling layers have even more symmetries than MLPs (Badrinarayanan et al., 2015). Given that in practice we cannot constraint the support of the posterior distribution to be the cone of identifiable parameter configurations and given that the likelihood model is also invariant under those transformations that do not change the function, the posterior landscape includes multiple equally likely modes that despite their different positions represent the same function. It is important to notice that this is always true for the likelihood but not for the posterior. Indeed, for the modes to be equally likely, the prior should also be invariant under those transformations, condition that is in general not true. Nevertheless, the fact that there are multiple modes of the posterior parametrizing for the same function remains true but they might be arbitrarily re-scaled by the prior[1]. As we will see in the following, this redundancy of the posterior is problematic when we want to obtain samples from it. Moreover it is interesting to notice how this issue disappears when the Bayesian inference is considered in the space of functions instead of weights. In this case, indeed, every optimal function has a unique mode in the landscape of the posterior and redundancy is not present:

$$\boldsymbol{f}_i \neq \boldsymbol{f}_j \implies p(\boldsymbol{f}_i|\mathcal{D}) \neq p(\boldsymbol{f}_j|\mathcal{D})\,.$$

In spite of that, performing inference over distributions of functions is prohibitive in practice due to the infinite dimensionality of the space in consideration. Only in very limited cases like the one of Gaussian Proccess, Bayesian inference is exact. Interestingly Neural network model in the limit of infinite width are Gaussian processes with a particular choice of the kernel determined by the architecture (Lee et al., 2017; Williams, 1998; Neal, 2012). In this limit Bayesian inference over functions can be performed analytically.

[1]Note that for the fully factorized Gaussian prior commonly adopted, the invariance under permutations is true

C. Quantify functional diversity

As illustrated in Section 2 , in the Bayesian context, predictions are made by doing a Monte-Carlo estimation of the BMA. Functional diversity and so the diversity in the hypotheses taken in consideration when performing the estimation, determines the epistemic uncertainty and the confidence over the predictions. Importantly, the epistemic uncertainty allows for the quantification of the likelihood of a test point to belong to the same distribution from which the training data points were sampled (Ovadia et al., 2019). Following this, the uncertainty can be used for the problem of Out-of-distribution (OOD) detection (Chen et al., 2020) that is often linked to the ability of a model to *"know what it doesn't know"*. A common way used in the literature to quantify the uncertainty is the Entropy[2] $\mathcal{H}$ of the predictive distribution:

$$\mathcal{H}\big\{p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\big\} = -\sum_{y} p(\mathbf{y}'|\mathbf{x}', \mathcal{D}) \log p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\,. \tag{10}$$

Nevertheless, it has been argued in recent works (Malinin et al., 2019) that this is not a good measure of uncertainty because it does not allow for a disentanglement of epistemic and aletoric uncertainty. Intuitively, we would like the predictive distribution of an OOD point to be uniform over the different classes. However, using the entropy and so the average prediction in the BMA, we are not able to distinguish between the case in which all the hypotheses disagree very confidently due to the epistemic uncertainty or are equally not confident due to the aleatoric uncertainty. To overcome this limitation, we can use a direct measure of the model disagreement computed as:

$$\mathcal{MD}^2(\mathbf{y}'; \mathbf{x}', \mathcal{D}) = \int_{\boldsymbol{w}} \big[p(\mathbf{y}'|\mathbf{x}', \boldsymbol{w}) - p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\big]^2 p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w}\,. \tag{11}$$

It is easy to see how the quantity in equation 11, measuring the deviation from the average prediction is zero when all models agree on the prediction. The latter can be the case of a training point where all hypotheses are confident or a noisy point where all models *"don't know"* the class and are equally uncertain. On the other side the model disagreement will be greater the zero the more the model disagree on a prediction representing like this the epistemic uncertainty. To obtain a scalar quantity out of equation 11 we can consider the expectation over the output space of $\mathbf{y}$:

$$\mathcal{MD}^2(\mathbf{x}') = \mathbb{E}_y\left[\int_{\boldsymbol{w}} \big[p(\mathbf{y}'|\mathbf{x}', \boldsymbol{w}) - p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\big]^2 p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w}\right]. \tag{12}$$

D. Functional derivative of the KL divergence

In this section, we show the derivation of the functional derivative for the KL divergence functional. We start with some preliminary definitions.

Given a manifold $\mathcal{M}$ embedded in $\mathbb{R}^d$, let $F[\rho]$ be a functional, i.e. a mapping from a normed linear space of function (Banach space) $\mathcal{F} = \{\rho(x) : x \in \mathcal{M}\}$ to the field of real numbers $F : \mathcal{F} \to \mathbb{R}$. The functional derivative $\delta F[\rho]/\delta\rho(x)$ represents the variation of value of the functional if the function $\rho(x)$ is changed.

Definition 2 (Functional derivative)**.** *Given a manifold $\mathcal{M}$ and a functional $F : \mathcal{F} \to \mathbb{R}$ with respect to ρ is defined as:*

$$\int \frac{\delta F}{\delta\rho(x)}\phi(x)dx = \lim_{\epsilon\to 0} \frac{F[\rho(x) + \epsilon\phi(x)] - F(\rho(x))}{\epsilon} = \frac{d}{d\epsilon} F[\rho(x) + \epsilon\phi(x)]\Big|_{\epsilon=0} \tag{13}$$

for every smooth ϕ.

Definition 3 (KL divergence)**.** *Given ρ and π two probability densities on $\mathcal{M}$, the KL-divergence is defined as:*

$$D_{KL}(\rho, \pi) = \int_{\mathcal{M}} \big(\log\rho(x) - \log\pi(x)\big)\rho(x)\, dx\,. \tag{14}$$

Proposition 1. *The functional derivative of the KL divergence in equation 14 is:*

$$\frac{\delta D_{KL}}{\delta\rho(x)} = \log\frac{\rho(x)}{\pi(x)} + 1 \tag{15}$$

[2]The continuous case is analogous using the differential entropy

Proof. using the definition of functional derivative in equation 13 :

$$\begin{aligned}\int \frac{\delta D_{KL}}{\delta \rho(x)}\phi(x)dx &= \frac{d}{d\epsilon}D_{KL}(\rho+\epsilon\phi,\pi)\Big|_{\epsilon=0} \\ &= \int \frac{d}{d\epsilon}\left[(\rho(x)+\epsilon\phi(x))\log\frac{(\rho(x)+\epsilon\phi(x))}{\pi(x)}\right]_{\epsilon=0} dx \\ &= \int \left[\phi(x)\log\frac{(\rho(x)+\epsilon\phi(x))}{\pi(x)} + \frac{d(\rho(x)+\epsilon\phi(x))}{d\epsilon}\right]_{\epsilon=0} dx \\ &= \int \left[\log\frac{\rho(x)}{\pi(x)}+1\right]\phi(x)dx\end{aligned} \tag{16}$$

□

E. Kernel density estimation

Kernel Density Estimation (KDE) is a nonparametric density estimation technique (Parzen, 1962). When an RBF kernel is used, it can be thought as a smoothed version of the empirical data distribution. Given some training datapoints $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ with $\mathbf{x}_i \sim p(\mathbf{x})$ and $\mathbf{x} \in \mathbb{R}^D$ their empirical distribution $q_0(\mathbf{x})$ is a mixture of n Dirac deltas centered at each training data:

$$q_0(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{x}_i)\,. \tag{17}$$

We can now smooth the latter by replacing each delta with an RBF kernel:

$$k_\epsilon(\mathbf{x},\mathbf{x}_i) = \frac{1}{h}\exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{h}\right) \tag{18}$$

where $h > 0$. The kernel density estimator is then defined as:

$$q_h(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}k_h(\mathbf{x},\mathbf{x}_i) \tag{19}$$

In the limit of $h \to 0$ and $N \to \infty$ the kernel density estimator is unbiased: it is equal to the true density. Indeed $k_{h\to 0}(\mathbf{x},\mathbf{x}_i) \to \delta(\mathbf{x}-\mathbf{x}_i)$ and so $q_{h\to 0}(\mathbf{x}) \to q_0(\mathbf{x})$ and:

$$\begin{aligned}\lim_{N\to\infty} q_0(\mathbf{x}) &= \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}(\delta(\mathbf{x}-\mathbf{x}_i)) \\ &= \mathbb{E}_{p(\mathbf{x}')}\left[\delta(\mathbf{x}-\mathbf{x}')\right] \\ &= \int_{\mathbb{R}^D}\delta(\mathbf{x}-\mathbf{x}')p(\mathbf{x}')\,d\mathbf{x}' = p(\mathbf{x})\end{aligned} \tag{20}$$

F. Comparison to Stein variational gradient descent

Note that our update is reminiscent of SVGD (Liu & Wang, 2016), which in parameter space can be written as:

$$\phi(\mathbf{w}_i^t) = \sum_{j=1}^{n}k(\mathbf{w}_i^t,\mathbf{w}_j^t)\nabla_{\mathbf{w}_i^t}\log p(\mathbf{w}_i^t|\mathcal{D}) + \sum_{j=1}^{n}\nabla_{\mathbf{w}_j^t}k(\mathbf{w}_j^t,\mathbf{w}_i^t)\,. \tag{21}$$

It is important to notice that here, the gradients are averaged across all the particles using the kernel matrix. Moving the inference from parameter to function space (Wang et al., 2019), leads to the update rule

$$\phi(\mathbf{w}_i^t) = \left(\frac{\partial \boldsymbol{f}_i^t}{\partial \mathbf{w}_i^t}\right)^\top\left(\frac{1}{n}\sum_{j=1}^{n}k(\boldsymbol{f}_i^t,\boldsymbol{f}_j^t)\nabla_{\boldsymbol{f}_j^t}\log p(\boldsymbol{f}_j^t|\mathcal{D}) + \nabla_{\boldsymbol{f}_j^t}k(\boldsymbol{f}_i^t,\boldsymbol{f}_j^t)\right). \tag{22}$$

This way of averaging gradients using a kernel can be dangerous in high-dimensional settings, where kernel methods often suffer from the curse of dimensionality. Moreover, in equation 21, the posterior gradients of the particles are averaged using their similarity in weight space, which can be misleading in multi-modal posteriors. Worse yet, in equation 22, the gradients are averaged in function space and are then projected back using exclusively the i-th Jacobian, which can be harmful given that it is not guaranteed that distances between functions evaluated on a subset of their input space resemble their true distance. Our proposed method, on the other hand, does not employ any averaging of the posterior gradients and thus comes closest to the true particle gradients in deep ensembles.

G. Gradient flow in function space

To theoretically justify the update rule introduced in function space in equation 4, we can rewrite the Liouville equation for the gradient flow in equation 6 in function space as

$$\begin{aligned}\frac{\partial \rho(\boldsymbol{f})}{\partial t} &= \nabla \cdot \left(\rho(\boldsymbol{f}) \nabla \frac{\delta}{\delta \rho} D_{KL}(\rho, \pi) \right) \\ &= \nabla \cdot \left(\rho(\boldsymbol{f}) \nabla \big(\log \rho(\boldsymbol{f}) - \log \pi(\boldsymbol{f}) \big) \right).\end{aligned} \tag{23}$$

Following this update, the mean field functional dynamics are

$$\frac{d\boldsymbol{f}}{dt} = -\nabla \big(\log \rho(\boldsymbol{f}) - \log \pi(\boldsymbol{f}) \big) . \tag{24}$$

Using the same KDE approximation as above, we can obtain a discretized evolution in function space and with it an explicit form for the repulsive force in equation 3 as

$$\boldsymbol{f}_{t+1}^i = \boldsymbol{f}_t^i + \epsilon_t \left(\nabla_{\boldsymbol{f}} \log \pi(\boldsymbol{f}_t^i) - \frac{\sum_{j=1}^n \nabla_{\boldsymbol{f}_i^t} k(\boldsymbol{f}_i^t, \boldsymbol{f}_j^t)}{\sum_{j=1}^n k(\boldsymbol{f}_i^t, \boldsymbol{f}_j^t)} \right) . \tag{25}$$

This gives a consistent formulation to specify α in equation 4 and equation 4 so that $\alpha \left(\{\boldsymbol{f}_j^t\}_{j=1}^n \right) := \sum_{j=1}^n k(\boldsymbol{f}_i^t, \boldsymbol{f}_j^t)$. The update rules using the SGE and SSGE approximations follow as for the parametric case.

H. SGE and SSGE repulsive forces

Recently, Li & Turner (2017) introduced the *Stein gradient estimator* (SGE) that offers better performance, while maintaining the same computational cost when compared to the KDE gradient estimator. Even more recently, Shi et al. (2018) introduced a spectral method for gradient estimation (SSGE), that also allows for a simple estimation on out-of-sample points. These two estimators can be used in equation 7, leading to the following update rules. The one using the Stein estimator, that we will call SGE-WGD, is:

$$x_i^{t+1} = x_i^t + \epsilon_t \left(\nabla \log \pi(x_i^t) + \sum_{j=1}^n (K + \eta \mathbb{I})_{ij}^{-1} \sum_{k=1}^n \nabla_{x_k^t} k(x_k^t, x_j^t) \right) , \tag{26}$$

where K is the kernel Gram matrix, η a small constant and $\mathbb{I}$ the identity matrix. We can notice the important difference between KDE and SGE that the former one is only considering the interaction of the i-th particle being updated with all the others, while the latter is simultaneously considering also the interactions between the remaining particles. The spectral method, that we will call SSGE-WGD, leads to the following update rule:

$$x_{t+1}^i = x_t^i + \epsilon_t \left(\nabla \log \pi(x_t^i) + \sum_{j=1}^J \frac{1}{\lambda_j^2} \sum_{m=1}^n \sum_{k=1}^n u_{jk} \nabla_{x_m} k(x_m^t, x_k^t) \cdot \sum_{l=1}^n u_{jl} k(x_i^t, x_l^t) \right) \tag{27}$$

where λ_j is the j-th eigenvalue of the kernel matrix and u_{jk} is the k-th component of the j-th eigenvector.

I. Sampling from synthetic distributions

As a sanity check, we first assessed the ability of our different approximations for Wasserstein gradient descent (using KDE, SGE, and SSGE) to sample from a two-dimensional univariate Gaussian distribution (Figure I.1). We see that our SGE-WGD

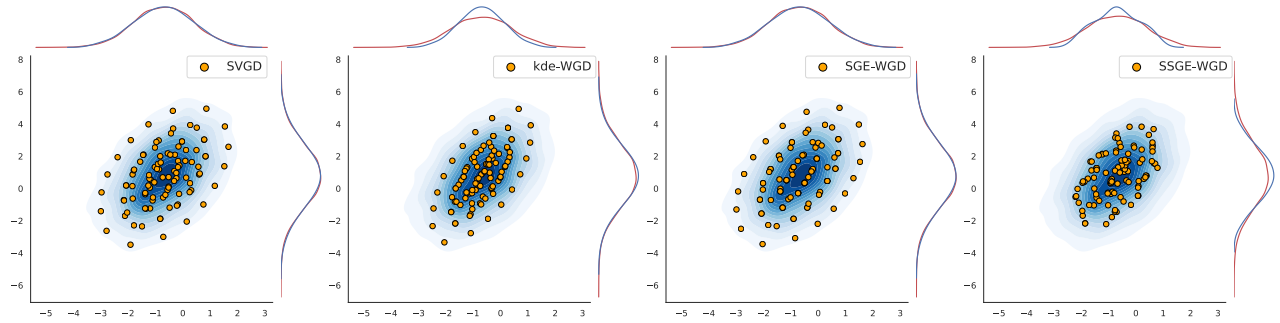

Figure I.1. **Single Gaussian.** We show samples from SVGD, KDE-WGD, SGE-WGD, and SSGE-WGD (from left to right). The SGE-WGD and SVGD fit the target almost perfectly.

and the SVGD fit the target almost perfectly. We also tested the different methods in a more complex two-dimensional Funnel distribution (Neal, 1995) $p(x, y) = \mathcal{N}(y|\mu = 0, \sigma = 3)\mathcal{N}(x|0, \exp(y/2))$, the results are reported in Figure I.2.. There, SGE-WGD and SVGD also perform best. In this section, we report the additional results for the different methods when sampling from the Funnel distribution

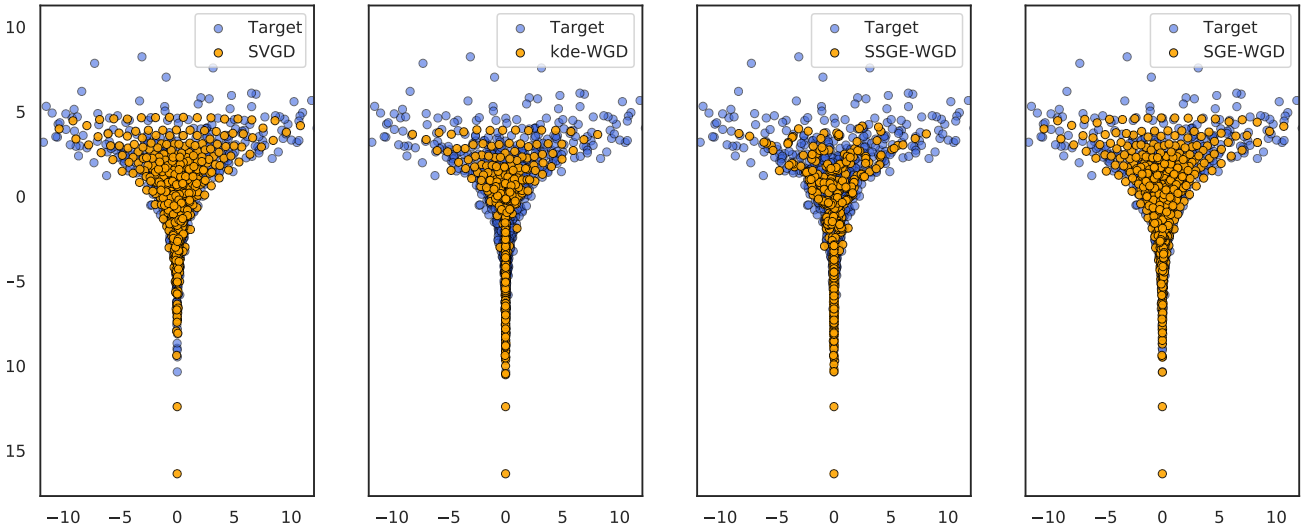

Figure I.2. **Neal's funnel.** The SGE-WGD and SVGD again fit the distribution best.

J. Implementation details

In this section, we report details on our implementation in the experiments we performed. All the experiments were performed on an internal cluster with NVIDIA GTX 1080 Ti and took roughly 150 GPU hours.

J.1. Sampling from synthetic distributions

Single Gaussian:we created a two-dimensional Gaussian distribution with mean $\mu = (-0.6871, 0.8010)$ and covariance $\Sigma = \begin{pmatrix} 1.130 & 0.826 \\ 0.826 & 3.389 \end{pmatrix}$. We used a normal initialization with zero mean and standard deviation $\sigma^2 = 3$. We sampled 100 initial particles and optimized them for 5000 iterations using Adam with a fixed learning rate of 0.1. The kernel bandwidth

was estimated using the median heuristic for all methods. For the SSGE we used all the eigenvalues. The random seed was fixed to 42.
Funnel: the target distribution followed the density $p(x, y) = \mathcal{N}(y|\mu = 0, \sigma = 3)\mathcal{N}(x|0, \exp(y/2))$. We used a normal initialization with zero mean and standard deviation $\sigma^2 = 3$. We sampled 500 initial particles and optimized them for 2000 iterations using Adam with a fixed learning rate of 0.1. The kernel bandwidth was fixed to 0.5 for all methods. For the SSGE we used all the eigenvalues. The random seed was fixed to 42.

J.2. 1D regression

We generated the training data by sampling 45 points from $x_i \sim \text{Uniform}(1.5, 2.5)$ and 45 from $x_i \sim \text{Uniform}(4.5, 6.0)$. The output y_i for a given x_i is then modeled following $y_i = x_i \sin(x_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, 0.25)$. We use a standard Gaussian likelihood and standard normal prior $\mathcal{N}(0, \mathbb{I})$. The model is a feed-forward neural network with 2 hidden layers and 50 hidden units with ReLU activation function. We use 50 particles initialized with random samples from the prior and optimize them using Adam (Kingma & Ba, 2014) with 15000 gradient steps, a learning rate of 0.01 and batchsize 64. The kernel bandwidth is estimated using the median heuristic. We tested the models on 100 uniformly distributed points in the interval $[0, 7]$. The random seed was fixed to 42.

J.3. 2D classification

We generate 200 training data points sampled from a mixture of 5 Gaussians with means equidistant on a ring of radius 5 and unitary covariance. The model is a feed-forward neural network with 2 hidden layers and 50 hidden units with ReLU activation function. We use a softmax likelihood and standard normal prior $\mathcal{N}(0, \mathbb{I})$. We use 100 particles initialized with random samples from the prior and optimize them using Adam (Kingma & Ba, 2014) with 10,000 gradient steps, a learning rate of 0.001 and batchsize 64. The kernel bandwidth is estimated using the median heuristic. The random seed was fixed to 42.

J.4. Classification on FashionMNIST

On this dataset, we use a feed-forward neural network with 3 hidden layers and 100 hidden units with ReLU activation function. We use a softmax likelihood and standard normal prior $\mathcal{N}(0, \mathbb{I})$. We use 50 particles initialized with random samples from the prior and optimize them using Adam (Kingma & Ba, 2014) for 50000 steps, a learning rate was 0.001 for sge-WGD,kde-WG,ssge-WGD and 0.0025 for kde-fWGD,ssge-fWGD,sge-fWGD, Deep ensemble, fSVGD, SVGD, and batchsize was 256. The kernel bandwidth is estimated using the median heuristic for all different methods. The learning rates were searched over the following values $(1e-4, 5e-4, 1e-3, 5e-3, 25e-4)$ we tested for 50000 and 30000 total number of iterations, 50 and 100 particles and batchsize 256 and 128. All results in Table 1 are averaged over the following random seeds $(38, 39, 40, 41, 42)$.

J.5. Classification on CIFAR-10

On this dataset, we used a residual network (ResNet32) with ReLU activation function. We use a softmax likelihood and standard normal prior $\mathcal{N}(0, 0.1\mathbb{I})$. We use 20 particles initialized using He initialization (He et al., 2015) and optimize them using Adam (Kingma & Ba, 2014) for 50000 steps, a learning rate was 0.00025 for sge-fWGD,kde-fWGD,ssge-fWGD,fSVGD and 0.0005 for kde-WGD,ssge-WGD,sge-fWGD, Deep ensemble and SVGD, and batchsize was 128. The kernel bandwidth is estimated using the median heuristic for all different methods. The learning rates were searched over the following values $(1e-4, 5e-4, 1e-3, 5e-3, 25e-4, 5e-5)$ we tested for 50000 and 30000 total number of iterations, 20 and 10 particles. All results in Table 1 are averaged over the following random seeds $(38, 39, 40, 41, 42)$.

K. Related Work and Future directions

The theoretical and empirical properties of SVGD have been well studied (Korba et al., 2020; Liu et al., 2019; D'Angelo & Fortuin, 2021) and it can also be seen as a Wasserstein gradient flow of the KL divergence in the Stein geometry (Duncan et al., 2019; Liu, 2017). Interestingly, a gradient flow interpretation is also possible for (stochastic gradient) MCMC-type algorithms (Liu et al., 2019), which can be unified under a general particle inference framework (Chen et al., 2018). Moreover, our Wasserstein gradient descent using the SGE approximation can also be derived using an alternative formulation as a gradient flow with smoothed test functions (Liu et al., 2019). A projected version of WGD has been studied

in (Wang et al., 2021), which could also be readily applied in our framework. Besides particle methods, Bayesian neural networks MacKay (1992); Neal (1995) have gained popularity recently (Wenzel et al., 2020a; Fortuin et al., 2021b; Fortuin, 2021; Izmailov et al., 2021), using modern MCMC (Neal, 1995; Wenzel et al., 2020a; Fortuin et al., 2021b; Garriga-Alonso & Fortuin, 2021; Fortuin et al., 2021a) and variational inference techniques (Blundell et al., 2015; Swiatkowski et al., 2020; Dusenberry et al., 2020; Immer et al., 2021a). Ensemble methods, on the other hand, have also been extensively studied (Lakshminarayanan et al., 2017; Fort et al., 2019; Wilson & Izmailov, 2020; Garipov et al., 2018) and many variants have been proposed (Wenzel et al., 2020b; He et al., 2020; Huang et al., 2017a; Zhang et al., 2019; Wen et al., 2020). Moreover, providing Bayesian interpretations for deep ensembles has been previously attempted through the lenses of stationary SGD distributions (Mandt et al., 2017; Chaudhari & Soatto, 2018), ensembles of linear models (Matthews et al., 2017), additional random functions (Ciosek et al., 2019; He et al., 2020), approximate inference (Wilson & Izmailov, 2020), and marginal likelihood lower bounds (Lyle et al., 2020). Furthermore, variational inference in function space has recently gained attention (Sun et al., 2019) and the limitations of the KL divergence have been studied in (Burt et al., 2020).

In future work, it will be interesting to study the impact of the Jacobian in the fWGD update and its implications on the Liouville equation in more detail, also compared to other neural network Jacobian methods, such as neural tangent kernels (Jacot et al., 2018) and generalized Gauss-Newton approximations (Immer et al., 2021b).Moreover, it would be interesting to derive explicit convergence bounds for our proposed method and compare them to the existing bounds for SVGD (Korba et al., 2020).