# On Stein Variational Neural Network Ensembles

**Francesco D'Angelo** [1] **Vincent Fortuin** [1] **Florian Wenzel** [2]

## Abstract

Ensembles of deep neural networks have achieved great success recently, but they do not offer a proper Bayesian justification. Moreover, while they allow for averaging of predictions over several hypotheses, they do not provide any guarantees for their diversity, leading to redundant solutions in function space. In contrast, particle-based inference methods, such as Stein variational gradient descent (SVGD), offer a Bayesian framework, but rely on the choice of a kernel to measure the similarity between ensemble members. In this work, we study different SVGD methods operating in the weight space, function space, and in a hybrid setting. We compare the SVGD approaches to other ensembling-based methods in terms of their theoretical properties and assess their empirical performance on synthetic and real-world tasks. We find that SVGD using functional and hybrid kernels can overcome the limitations of deep ensembles. It improves on functional diversity and uncertainty estimation and approaches the true Bayesian posterior more closely. Moreover, we show that using stochastic SVGD updates, as opposed to the standard deterministic ones, can further improve the performance.

## 1. Introduction

Ensembling methods for neural networks have 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). It has even been argued that these methods can be viewed as an approximate way of performing Bayesian inference in neural networks (Wilson & Izmailov, 2020). However, while they do allow for the averaging of predictions over several hypotheses, they do not encourage diversity between hypotheses, in contrast to actual Bayesian neural networks (BNNs) (MacKay, 1992). One way to bridge the gap between deep ensembles and BNNs is through Stein variational gradient descent (SVGD) (Liu & Wang, 2016). SVGD uses particles to approximate the Bayes posterior and therefore in practice also trains an ensemble of neural network models. However, as opposed to standard deep ensembles, it uses an objective function that encourages diversity in the ensemble through the use of a kernel. Moreover, it guarantees asymptotic convergence to the true posterior (Liu, 2017; Korba et al., 2020). While SVGD can naïvely be defined using a kernel in the weight space of the neural network, this can lead to suboptimal results in the case of overparametrized models like BNNs. For instance, several neural network particles could have very different weights, but implement the exact same function, thus giving a false sense of diversity in the ensemble (Badrinarayanan et al., 2015; Fort et al., 2019). Moreover, it has recently been proposed that a stochatic SVGD can improve the diversity even further (Gallego & Insua, 2018).

In this work, we explore different Stein-based ensembling methods and assess their diversity between predictions. We compare SVGD methods on a spectrum between pure weight-space and pure function-space inference, using deterministic and stochastic updates. We also include two new approaches in our comparison and show that our hybrid h-SVGD method, that acts both in the weight and function space, and our fw-SVGD method, lead to more diverse ensembles and improved uncertainty estimation and out-of-distribution detection. We make the following contributions:

- We describe a generalized framework for Stein ensembles, which includes existing methods (such as standard deep ensembles) and study different kernel choices in the deterministic and stochastic setting.
- We empirically compare the different methods on several synthetic and real-world data sets and highlight their strengths and weaknesses for various use cases.
- We find that SVGD variants with function-space kernels, including our newly proposed ones, can empirically lead to improved uncertainty estimation and out-of-distribution detection.
- We also find that stochastic updates can improve performance over deterministic ones.

[1] ETH Zürich, Zürich, Switzerland [2]Google Research, Berlin, Germany. Correspondence to: Francesco D'Angelo <fdangelo@student.ethz.ch>.

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

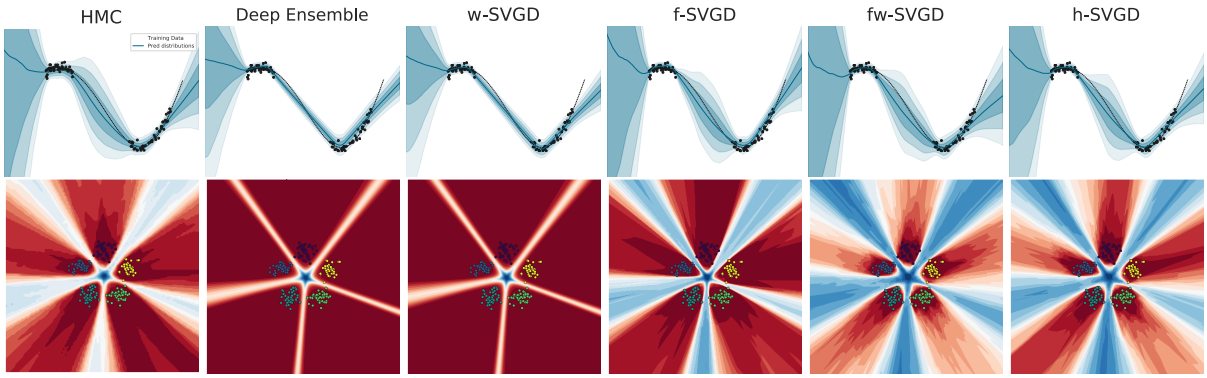



*Figure 1.* In the regression (top), the shaded areas represent the standard deviation and in the classification setting (bottom), the entropy of the predictive posteriors. Deep ensembles and SVGD in weight space (w-SVGD) do not encourage diversity in the function space and fail to represent in-distribution uncertainty in regression and uncertainty away from the training data in classification. The functional methods (f-SVGD and our new methods fw-SVGD and h-SVGD) improve diversity and uncertainty representation and closely approach the ground truth posterior approximated by HMC.

## 2. Background

**Deep Neural Network Ensembles** Ensembles of neural networks have a long history (e.g., Levin et al., 1990; Hansen & Salamon, 1990; Breiman, 1996) and were recently revisited by Lakshminarayanan et al. (2017) and coined *deep ensembles*. A deep ensemble is constructed by training several networks *independently* by maximizing the posterior and averaging their predictions.

**Bayesian Neural Networks** In supervised deep learning, we typically consider a likelihood function $p(\boldsymbol{y}|f(\boldsymbol{x};\boldsymbol{w}))$ (e.g., Gaussian for regression or Categorical for classification) parameterized by a neural network $f(\boldsymbol{x};\boldsymbol{w})$ and training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$. In Bayesian neural networks (BNNs), we are interested in the posterior distribution of all likely networks given by $p(\boldsymbol{w}|\mathcal{D}) \propto \prod_{i=1}^n p(\boldsymbol{y}_i|f(\boldsymbol{x}_i;\boldsymbol{w}))\, p(\boldsymbol{w})$, where $p(\boldsymbol{w})$ 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 $\widehat{\boldsymbol{w}}$ to predict $\boldsymbol{y}^* = f(\boldsymbol{x}^*;\widehat{\boldsymbol{w}})$, but we marginalize over the whole posterior, thus taking all possible explanations of the data into account:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|f(\boldsymbol{x}^*;\boldsymbol{w}))\, p(\boldsymbol{w}|\mathcal{D})\, \mathrm{d}\boldsymbol{w} \tag{1}$$

The main challenge with BNNs is that the posterior can usually only be computed up to a constant factor.

## 3. Stein Ensembles

In the following, we present our framework for Stein ensembles. We will consider a set of $n$ particles $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$ and evolve them according to a specific update rule $\boldsymbol{w}_i^{t+1} \leftarrow \boldsymbol{w}_i^t + \epsilon_t \phi(\boldsymbol{w}_i^t)$. These particles can be used to approximate Eq. (1) by $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \frac{1}{n}\sum_{i=1}^n p(\boldsymbol{y}^*|f(\boldsymbol{x}^*;\boldsymbol{w}_i))$.

**Weight-kernel weight SVGD (w-SVGD)** When applied naïvely to BNN inference, the original SVGD method (Liu & Wang, 2016) (and its stochastic variant (Gallego & Insua, 2018)) perform the inference directly in the weight space, using a positive definite kernel that is also defined in the weight space (Hu et al., 2019). We denote this method by (w-SVGD) and show the full update rule here:

$$\phi(\boldsymbol{w}_i) = \frac{1}{n}\sum_{j=1}^n \Bigg( \underbrace{k(\boldsymbol{w}_j, \boldsymbol{w}_i)\nabla_{\boldsymbol{w}_j} \log p(\boldsymbol{w}_j|\mathcal{D})}_{\text{driving force}} + \underbrace{\nabla_{\boldsymbol{w}_j} k(\boldsymbol{w}_j, \boldsymbol{w}_i)}_{\text{repulsive force}} \Bigg) + \beta \sum_{j=1}^n \sqrt{\frac{2\mathcal{K}_{ij}}{\epsilon_t}}\, \eta_j \tag{2}$$

with $\mathcal{K}_{ij} = \frac{1}{n} k(\boldsymbol{w}_i, \boldsymbol{w}_j)\mathbb{I}_{d\times d}$ and $\eta_j \sim \mathcal{N}(0, \mathbb{I}_{d\times d})$. This update rule is deterministic with $\beta = 0$ and stochastic with $\beta = 1$. The driving force moves the particles towards areas of higher posterior probability mass by following the gradient of the log posterior. Through the kernel-weighted average of the gradients, the particles can inform each other about high-probability regions in the space. The repulsive force, instead, pushes the particles away from each other, avoiding collapse to a posterior mode and encouraging weight diversity. The noise term with $\beta = 1$ plays a key role, as it is designed to make the posterior invariant under the dynamics and thus ensures that the Markov chain evolving the single particles, *even with a finite number of particles*, converges to the posterior as $t \to \infty$.

**SVGD as Deep ensembles (DE) and SGLD** When using the indicator function as a kernel, that is $k(\boldsymbol{w}_i, \boldsymbol{w}_j) = n\,\mathbb{1}(\boldsymbol{w}_i, \boldsymbol{w}_j)$, the w-SVGD simplifies to the standard deep ensemble, reducing for $\beta = 0$ to MAP estimations:

$$\phi(\boldsymbol{w}_i) = \nabla_{\boldsymbol{w}_i} \log p(\boldsymbol{w}_i|\mathcal{D}) + \beta\sqrt{\frac{2}{\epsilon_t}}\,\eta_i \tag{3}$$

With $\eta_i \sim \mathcal{N}(0, \mathbb{I})$. We can see immediately that this removes any repulsion and thus does not enforce diversity in the ensemble. Moreover, for $\beta = 1$, we recover the update rule for the SGLD algorithm (Welling & Teh, 2011).

**Kernel in function space** Despite the convergence properties of SVGD in the asymptotic limit, there is empirical (Zhuo et al., 2018) and theoretical (Zhang et al., 2020) evidence that, in the finite regime $n \ll \infty$, the particles tend to collapse to a few local modes. Moreover, the over-parameterized nature of BNN models leads to a target posterior which has a large number of modes that—despite their distance in the weight space—parameterize the same function (Badrinarayanan et al., 2015). Predictive distributions approximated by samples from these modes do not improve over a simple point estimate and lead to a poor uncertainty estimation. To overcome the above limitations of deep ensembles, as well as naïve SVGD, we investigate the different possibilities of incorporating the knowledge of the parameterized functions into the update rule of SVGD. To this end, we aim to use the repulsion in Eq. (2) with a kernel acting in function space instead of the weight space to directly encourage functional diversity of the particles and avoid functional posterior collapse. Let $g : (\boldsymbol{x}, \boldsymbol{w}) \mapsto f(\boldsymbol{x}; \boldsymbol{w})$ be the map that maps a data point $\boldsymbol{x} \in \mathcal{X}$ and weight vector $\boldsymbol{w}$ to the corresponding neural network output. We then denote the neural network outputs with weight $\boldsymbol{w}_i$ as $\boldsymbol{f}_i := f(\boldsymbol{x}; \boldsymbol{w}_i)$. If we now define a kernel $k_f(\cdot, \cdot)$ over these function outputs (e.g., an RBF kernel), we can map it to a new kernel $k_w$ in the weight space as $k_w = k_f \circ g$. That is, $k_w(\boldsymbol{w}_i, \boldsymbol{w}_j) = k_f\left(g(\boldsymbol{x}, \boldsymbol{w}_i), g(\boldsymbol{x}, \boldsymbol{w}_j)\right)$. Note that this is a valid kernel due to the compositionality of kernels (Bishop, 2006, eq. 6.19). Thus, we can use $k_w$ for weight-space inference using SVGD, while still taking functional diversity into account through the dependence on $k_f$. The map $g(\boldsymbol{x}, \cdot)$ is not injective due to the over-parameterization, therefore the kernel defined with it is only positive-semidefinite. Hence, the asymptotic convergence properties of SVGD are not met. However from a practical standpoint, the repulsive component avoids redundancies in the functions and hence preserve the non-singularity of the matrix. At any rate, the main attention of our work is not on the quality of the weight posterior approximation, but on an efficient characterization of the functional posterior and the training of a functionally diverse ensemble.

**The functional SVGD update rule** We can use the kernel in function space to develop different update rules based on the original formulation in Eq. (2). We consider the implicit functional likelihood $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{f})$, as well as the functional prior $p(\boldsymbol{f})$, which can either be defined separately or as the push-forward measure of a weight space prior $p(\boldsymbol{w})$. In the following, we discuss the advantages and disadvantages of the different choices and include two novel approaches (fw-SVGD and h-SVGD).

**Function-kernel weight SVGD (fw-SVGD)** The simplest solution to take functional diversity into account and operate on functions instead of the weights, is to use the functional kernel, and substitute it in Eq. (2):

$$\phi(\boldsymbol{w}_i) = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{J}_j^\top \Big( k(\boldsymbol{f}_i, \boldsymbol{f}_j) \nabla_{\boldsymbol{f}_j} \log p(\boldsymbol{f}_j|\mathcal{D}) + \nabla_{\boldsymbol{f}_j} k(\boldsymbol{f}_i, \boldsymbol{f}_j) \Big) + \beta \sum_{j=1}^{n} \sqrt{\frac{2\mathcal{K}_{ij}}{\epsilon_t}}\, \eta_j \tag{4}$$

where $\mathcal{K}_{ij} = \frac{1}{n} k(\boldsymbol{f}_i, \boldsymbol{f}_j) \mathbb{I}_{d\times d}$ and $\boldsymbol{J}_j = \partial \boldsymbol{f}_j / \partial \boldsymbol{w}_j$ denotes the Jacobian of $\boldsymbol{f}_j$ with respect to $\boldsymbol{w}_j$, that can be interpreted as a projector that maps the updates from the function space into the weight space. Intuitively, the main issue with this approach is that the functional kernel might be a good measure for functional diversity, but a bad similarity measure for averaging the log-posterior gradients in the weight space. As mentioned before, similar functions might be implemented by very different weights and may therefore share gradients in this approach even though they are far away from each other in the weight space. However, the method can still satisfy convergence under some assumptions: for the deterministic case ($\beta = 0$), the minimization of the KL divergence can be ensured provided that the functional kernel belongs to the Stein class. For the stochastic case ($\beta = 1$), a weaker assumption is sufficient to have convergence to the posterior in the asymptotic limit: the kernel Gram matrix of the functional kernel is positive definite at each step, such that Theorem 1 in Appendix A.2 holds.

**Functional SVGD (f-SVGD)** A similar version of the previous update rule was introduced by Wang et al. (2019b):

$$\phi(\boldsymbol{w}_i) = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{J}_i^\top \Big( k(\boldsymbol{f}_i, \boldsymbol{f}_j) \nabla_{\boldsymbol{f}_j} \log p(\boldsymbol{f}_j|\mathcal{D}) + \nabla_{\boldsymbol{f}_j} k(\boldsymbol{f}_i, \boldsymbol{f}_j) \Big). \tag{5}$$

One major issue with this approach, compared to the previous ones, is the requirement of a function space prior gradient: $\nabla_{\boldsymbol{f}_j} \log p(\boldsymbol{f}_j|\boldsymbol{x}, \boldsymbol{y}) = \nabla_{\boldsymbol{f}_j} \log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{f}_j) + \nabla_{\boldsymbol{f}_j} \log p(\boldsymbol{f}_j)$. This can be estimated using Shi et al. (2018), since the implicit prior in function space defined by the one in weight space is not analytically accessible (Sun et al., 2019). Alternatively, a Gaussian process (GP) prior can be used. Crucially, Eq. (5) only uses the Jacobian matrix of particle $i$ which we aim to update. This difference is fundamental and leads to completely different inference dynamics, since in the f-SVGD update, all gradients are computed in the function space and are then projected back using exclusively the $i$-th Jacobian. In contrast, the fw-SVGD update in Eq. (4) uses a separate Jacobian $\boldsymbol{J}_j$ for each gradient $\nabla_{\boldsymbol{f}_j}$ and thus calculates an expectation in the weight space

| | | FashionMNIST | | | | | | CIFAR10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **AUROC(H)** | **AUROC(MD)** | **Accuracy** | $\mathbf{MD_o/MD_t}$ | **ECE** | **NLL** | **AUROC(H)** | **Accuracy** | $\mathbf{MD_o/MD_t}$ | **ECE** | **NLL** |
| $\beta = 0$ | **DE** | 0.938±0.001 | 0.977±0.001 | 88.864±0.015 | 7.566±0.011 | 0.013±0.001 | 0.179±0.001 | 0.854±0.004 | **85.434±0.067** | 1.527±0.012 | **0.064±0.001** | **0.296±0.002** |
| | **w-SVGD** | 0.941±0.001 | 0.978±0.001 | 88.280±0.041 | 7.422±0.010 | 0.017±0.001 | 0.191±0.001 | 0.838±0.004 | 85.338±0.048 | 1.478±0.011 | 0.065±0.001 | 0.299±0.001 |
| | **f-SVGD** | **0.974±0.001** | 0.979±0.001 | 87.780±0.101 | 7.767±0.023 | 0.067±0.001 | 0.262±0.003 | 0.782±0.002 | 81.948±0.041 | 1.467±0.001 | 0.208±0.003 | 0.311±0.001 |
| | **fw-SVGD-l** | 0.948±0.001 | 0.979±0.001 | 88.792±0.020 | 7.303±0.004 | 0.021±0.001 | 0.190±0.001 | 0.852±0.003 | 85.234±0.040 | 1.459±0.007 | 0.071±0.001 | 0.307±0.001 |
| | **fw-SVGD-s** | 0.958±0.001 | **0.986±0.001** | 88.510±0.034 | 7.783±0.036 | **0.010±0.001** | 0.182±0.001 | 0.843±0.002 | 85.364±0.018 | 1.409±0.012 | 0.070±0.001 | 0.304±0.001 |
| | **h-SVGD-l** | 0.959±0.001 | **0.985±0.001** | **89.080±0.034** | **7.835±0.018** | **0.011±0.001** | **0.173±0.001** | 0.853±0.002 | 85.230±0.098 | 1.481±0.007 | **0.064±0.001** | 0.300±0.001 |
| | **h-SVGD-s** | 0.931±0.001 | 0.969±0.001 | 88.870±0.015 | 7.576±0.019 | 0.016±0.001 | 0.183±0.001 | **0.861±0.002** | 85.250±0.049 | **1.537±0.004** | 0.069±0.008 | 0.303±0.001 |
| $\beta = 1$ | **pSGLD** | 0.902±0.001 | 0.957±0.001 | 88.562±0.049 | 6.974±0.016 | 0.022±0.001 | 0.194±0.001 | 0.861±0.002 | 85.818±0.035 | 1.517±0.005 | **0.065±0.001** | 0.290±0.001 |
| | **w-SVGD** | 0.925±0.001 | 0.971±0.001 | 87.390±0.032 | 7.335±0.027 | 0.018±0.001 | 0.208±0.001 | 0.878±0.002 | 86.720±0.073 | 1.540±0.001 | 0.223±0.004 | 0.275±0.002 |
| | **fw-SVGD-l** | 0.951±0.001 | 0.982±0.001 | **88.876±0.035** | 7.442±0.020 | 0.021±0.001 | 0.190±0.001 | 0.844±0.005 | 85.754±0.079 | 1.448±0.005 | 0.254±0.008 | 0.306±0.001 |
| | **fw-SVGD-s** | **0.955±0.001** | **0.984±0.001** | 88.616±0.027 | **7.718±0.021** | **0.013±0.001** | **0.183±0.001** | 0.852±0.003 | 85.284±0.096 | 1.407±0.006 | 0.246±0.009 | 0.308±0.002 |
| | **h-SVGD-l** | 0.929±0.001 | 0.974±0.001 | 87.524±0.036 | 7.253±0.031 | 0.023±0.001 | 0.211±0.001 | 0.880±0.001 | **86.862±0.056** | 1.580±0.001 | 0.201±0.003 | 0.270±0.001 |
| | **h-SVGD-s** | 0.933±0.001 | 0.975±0.001 | 87.730±0.031 | 7.393±0.010 | 0.025±0.001 | 0.210±0.001 | **0.890±0.002** | 86.666±0.037 | **1.600±0.002** | 0.220±0.006 | **0.269±0.001** |

*Table 1.* **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 for model disagreement on OOD and test points. **FashionMNIST :** for $\beta = 0$, we see that the h-SVGD-l performs best in terms of accuracy, NLL, and OOD detection using MD and leads to the most functionally diverse and calibrated ensemble. The f-SVGD instead achieves the best OOD detection using the entropy. $\beta = 1$: The fw-SVGD-l achieves the best accuracy, but the fw-SVGD-s yields the best OOD detection using both entropy and MD. **CIFAR10:** for $\beta = 0$, the DE performs best in terms of accuracy and likelihood, while the h-SVGD-s offers better OOD detection and functional diversity. The f-SVGD instead shows the worst results in terms of both OOD detection and accuracy. $\beta = 1$: The h-SVGD-l performs best in terms of accuracy, while the h-SVGD-s offers better OOD detection and functional diversity.

instead of the function space. Intuitively, this means that the f-SVGD approximates the map $g$ from weight to function space with one global linear function, extrapolated from the single point $\boldsymbol{f}_i$, while the fw-SVGD approximates $g$ locally at each point $\boldsymbol{f}_j$ with $\boldsymbol{J}_j$.

**Hybrid-kernel weight SVGD (h-SVGD)** Finally, we consider another novel update rule that combines a kernel in the weight space, to correctly average the posterior gradient contributions of the different particles, with a functional kernel, to repel particles parameterizing the same function and thus encourage functional diversity:

$$\phi(\boldsymbol{w}_i) = \frac{1}{n}\sum_{j=1}^{n}\Big(k(\boldsymbol{w}_i, \boldsymbol{w}_j)\nabla_{\boldsymbol{w}_j}\log p(\boldsymbol{w}_j|\mathcal{D}) + \nabla_{\boldsymbol{w}_j} k(\boldsymbol{f}_i, \boldsymbol{f}_j)\Big) + \beta\sum_{j=1}^{n}\sqrt{\frac{2\mathcal{K}_{ij}}{\epsilon_t}}\,\eta_j \tag{6}$$

Since this update rule uses two different kernels, it does not technically constitute a version of the original SVGD, and thus does not inherit its strong convergence guarantees. Moreover, for the stochastic version ($\beta = 1$), it is not possible to guarantee that the stationary distribution is uniquely the posterior. However, we find that this constitutes a useful tradeoff between weight-space and function-space methods and yields ensemble members with performant functions, while still encouraging functional diversity.

## 4. Experiments

We evaluate the different Stein ensemble methods described above and compare them against deep ensembles and HMC. At first qualitatively on a synthetic regression and a classification task reported in Figure 1. We then evaluate the predictive performance on standard real-world classification datasets and assess the uncertainty estimation of the methods in terms of calibration and out-of-distribution (OOD) detection. To assess the diversity of the ensemble generated by the different methods in function space, we measure the functional diversity using the model disagreement (MD) (details in Appendix A.1). In all experiments, the letter **-l** added next to a method indicates that the functional kernel is evaluated on the logits, whereas the letter **-s** indicates the evaluation on the softmax outputs.

**Image classification** First, we test the methods using the FashionMNIST dataset (Xiao et al., 2017) for training and the MNIST dataset (LeCun) as OOD data; results are reported in Table 1 (left). Secondly, we use a residual network (ResNet32) architecture (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009). Here, we use the SVHN dataset (Netzer et al., 2011) as OOD data. The results are reported in Table 1 (right). Additional results are reported in Appendix B.2 and in Appendix B.3.

## 5. Conclusion

We have presented a framework for Stein ensembling methods for deep neural networks. We have shown that incorporating functional repulsion between ensemble members is non-trivial, but that it can improve the quality of the estimated uncertainties and OOD detection on synthetic and real-world data and seems to approach the true Bayesian posterior more closely. We proposed previously undescribed SVGD variants (fw-SVGD and h-SVGD) that can outperform the standard methods on some tasks and have shown that stochastic SVGD can be reformulated with functional repulsion and can outperform its deterministic counterparts.

## References


Badrinarayanan, V., Mishra, B., and Cipolla, R. Understanding symmetries in deep networks. *arXiv preprint arXiv:1511.01029*, 2015.

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

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.

Chang, W.-C., Li, C.-L., Mroueh, Y., and Yang, Y. Kernel stein generative modeling. *arXiv preprint arXiv:2007.03074*, 2020.

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

Chen, P. and Ghattas, O. Projected stein variational gradient descent. *arXiv preprint arXiv:2002.03469*, 2020.

Chewi, S., Gouic, T. L., Lu, C., Maunu, T., and Rigollet, P. Svgd as a kernelized wasserstein gradient flow of the chi-squared divergence. *arXiv preprint arXiv:2006.02509*, 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.

Detommaso, G., Cui, T., Marzouk, Y., Spantini, A., and Scheichl, R. A stein variational newton method. In *Advances in Neural Information Processing Systems*, pp. 9169–9179, 2018.

Duncan, A., Nüsken, 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.

Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., and Carin, L. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*, 2019.

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

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

Gorham, J., Raj, A., and Mackey, L. Stochastic stein discrepancies. *arXiv preprint arXiv:2007.02857*, 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, 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.

Hu, X., Szerlip, P., Karaletsos, T., and Singh, R. Applying svgd to bayesian neural networks for cyclical time-series prediction and inference. *arXiv preprint arXiv:1901.05906*, 2019.

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*, 2021.

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.

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/*.

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.

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. *Advances in neural information processing systems*, 29:2378–2386, 2016.

Ma, Y.-A., Chen, T., and Fox, E. B. A complete recipe for stochastic gradient mcmc. *arXiv preprint arXiv:1506.04696*, 2015.

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., McInerney, J., Abrol, F., Ranganath, R., and Blei, D. Variational tempering. In *Artificial Intelligence and Statistics*, pp. 704–712, 2016.

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

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

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.

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.

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, D., Tang, Z., Bajaj, C., and Liu, Q. Stein variational gradient descent with matrix-valued kernels. In *Advances in neural information processing systems*, pp. 7836–7846, 2019a.

Wang, Z., Ren, T., Zhu, J., and Zhang, B. Function space particle optimization for bayesian neural networks. *arXiv preprint arXiv:1902.09754*, 2019b.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Wenzel, F., Roth, K., Veeling, B. S., 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 of Machine Learning*, 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.

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.

Yao, Y., Vehtari, A., and Gelman, A. Stacking for non-mixing bayesian computations: The curse and blessing of

multimodal posteriors. *arXiv preprint arXiv:2006.12335*, 2020.

Ye, M., Ren, T., and Liu, Q. Stein self-repulsive dynamics: Benefits from past samples. *arXiv preprint arXiv:2002.09070*, 2020.

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

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.

Zhuo, J., Liu, C., Shi, J., Zhu, J., Chen, N., and Zhang, B. Message passing stein variational gradient descent. In *International Conference on Machine Learning*, pp. 6018–6027. PMLR, 2018.

# Supplementary Material: Stein Ensembles

## A. Additional definitions

### A.1. Quantify functional diversity

As illustrated in Eq. (1), 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 where 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[1] $\mathcal{H}$ of the predictive distribution:

$$\mathcal{H}\big\{p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\big\} = -\sum_{y} p(\mathbf{y}'|\mathbf{x}', \mathcal{D}) \log p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\,. \tag{7}$$

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:

$$\mathcal{MD}^2(\mathbf{y}'; \mathbf{x}', \mathcal{D}) = \int_{\boldsymbol{w}} \big[p(\mathbf{y}'|\mathbf{x}', \boldsymbol{w}) - p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\big]^2 p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w}\,. \tag{8}$$

It is easy to see how the quantity in Eq. (8), 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 Eq. (8) we can consider the expectation over the output space of $\mathbf{y}$:

$$\mathcal{MD}^2(\mathbf{x}') = \mathbb{E}_y\left[\int_{\boldsymbol{w}} \big[p(\mathbf{y}'|\mathbf{x}', \boldsymbol{w}) - p(\mathbf{y}'|\mathbf{x}', \mathcal{D})\big]^2 p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w}\right]. \tag{9}$$

### A.2. Stochastic SVGD convergence

Interestingly, (Gallego & Insua, 2018) noticed how the stochastic update rule in Eq. (2) can be framed in the framework developed in (Ma et al., 2015) by considering an augmented space given by the concatenation of the particles $\tilde{x} = (x_1, \ldots, x_n) \in \mathbb{R}^{nd}$ to obtain a valid MCMC-type posterior sampler that naturally runs $n$ interacting Markov chains. With this design the method falls into the class of settings for which Theorem 1 in Ma et al. (2015) is valid; with the condition that the matrix $\mathbf{D}$, and so the Gram matrix of the kernel in this case, is positive definite:

**Theorem 1** (Convergence of stochastic SVGD (Gallego & Insua, 2018))**.** *Assuming a positive definite kernel Gram matrix with elements $k(\boldsymbol{w}_j, \boldsymbol{w}_i)$, the discretization of the stochastic SVGD SDE in Eq. (2) has the product measure of the target $\tilde{p}(\tilde{\boldsymbol{w}}) = \prod_{j=1}^{n} p(\boldsymbol{w}_j)$ as a unique stationary distribution. Convergence is ensured in the asymptotic limit of the step size $\epsilon_t \to 0$ under the Robbins-Monro conditions.*

## B. Additional experimental results

In this section, we show the results for the stochastic versions on the synthetic data and complementary results for FashionMNIST and CIFAR-10.

### B.1. Low-dimensional synthetic problems

In Figure B.1 we report the results for the stochastic methods for the one-dimensional regression task and the two-dimensioanl classification.

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

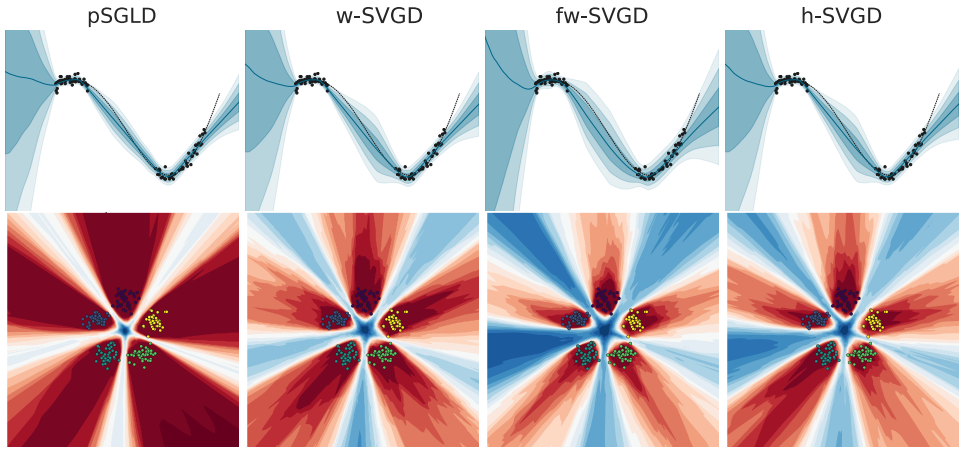



*Figure B.1.* **Synthetic data, $\beta = 1$.** Predictive posteriors for the different stochastic update methods on a synthetic 1D regression (top) and 2D classification (bottom) task. Shaded areas in the regression represent the standard deviation of the predictive posteriors, for the classification we show the entropy of the predictive posteriors. We see again that the function-space methods capture the uncertainty better than the weight-space ones, but all methods generally seem to capture it better than in the case of using deterministic updates.

## B.2. Calibration curves

In this section, following the protocol of (Ciosek et al., 2019), we show the calibration curves for the different methods on FashionMNIST in Figure B.2 and on CIFAR10 in Figure B.3. These curves show the average accuracy of all methods on differently sized subsets of a test dataset, which is a combination of the (FashionMNIST,CIFAR10) test set and 10,000 OOD (MNIST,SVHM) data points. Each subset is constructed by including the $k$ points with the smallest uncertainty as predicted by the respective method. In the plots we display only the first 10,000 points sorted by confidence, to make the results better visible. This simulates a setting in which the model is deployed on data that may or may not be OOD and has to decide based on its uncertainty estimates for which points to actually make a prediction. If a model is well-calibrated, this curve should decrease monotonically when increasing the subset size.

### B.2.1. FashionMNIST

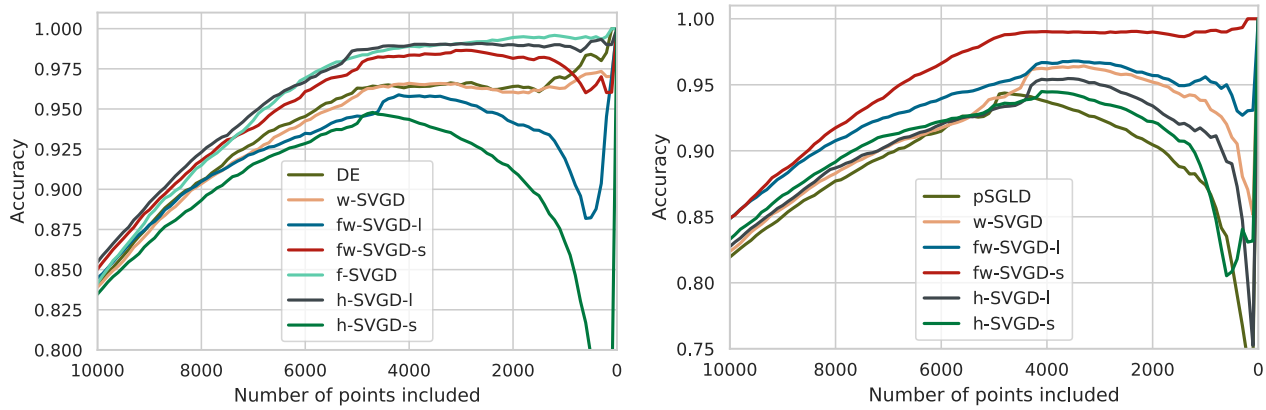



*Figure B.2.* **Calibration curves FashionMNIST:** calibration curves showing the relationship between uncertainty (horizontal axis) and accuracy (vertical axis) for the stochastic methods $\beta = 1$ (right) and deterministic $\beta = 0$(left).

We can see in Figure B.2 that for f-SVGD, h-SVGD-l, w-SVGD and DE the curve is monotonically decreasing. Conversely h-SVGD-s, fw-SVGD-l and fw-SVGD-s do not display this behaviour as the accuracy is decreasing between 5000 and 0, fact that reflects low uncertainty over OOD points that are then incorrectly classified. Furthermore, the area under this curve can be seen as a combined measure for accuracy and uncertainty calibration. For instance, when setting a target accuracy of

97 %, the h-SVGD-l and f-SVGD in Figure B.2 could be used to classify 6,000 points which is more than for any of the other methods. For the stochastic case instead, fw-SVGD-s shows the best performance and appears to be the only calibrated method.

#### B.2.2. CIFAR10

We observe in Figure B.3 that on CIFAR10, the h-SVGD-l appears to be the one with the largest area under the curve and so the one achieving the best tradeoff between uncertainty and accuracy in both the deterministic and stochastic case.

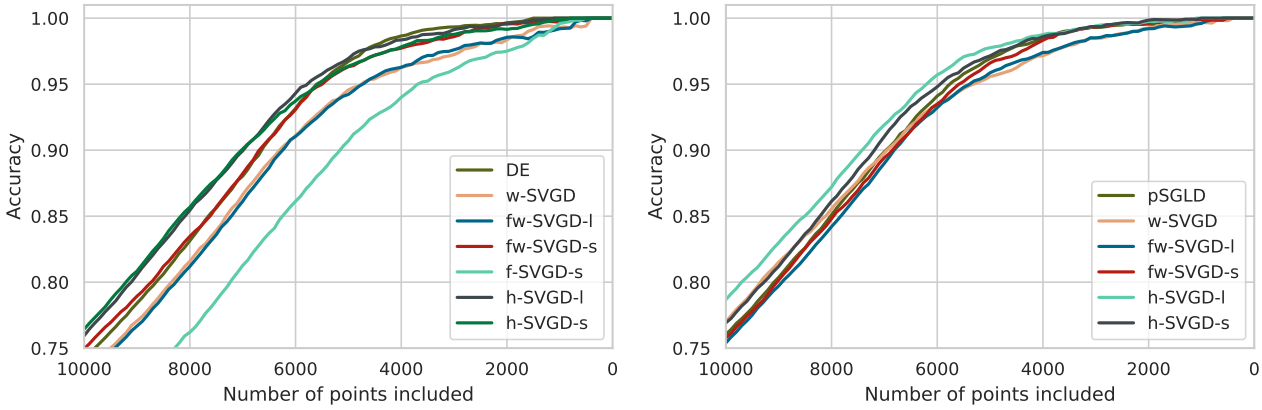
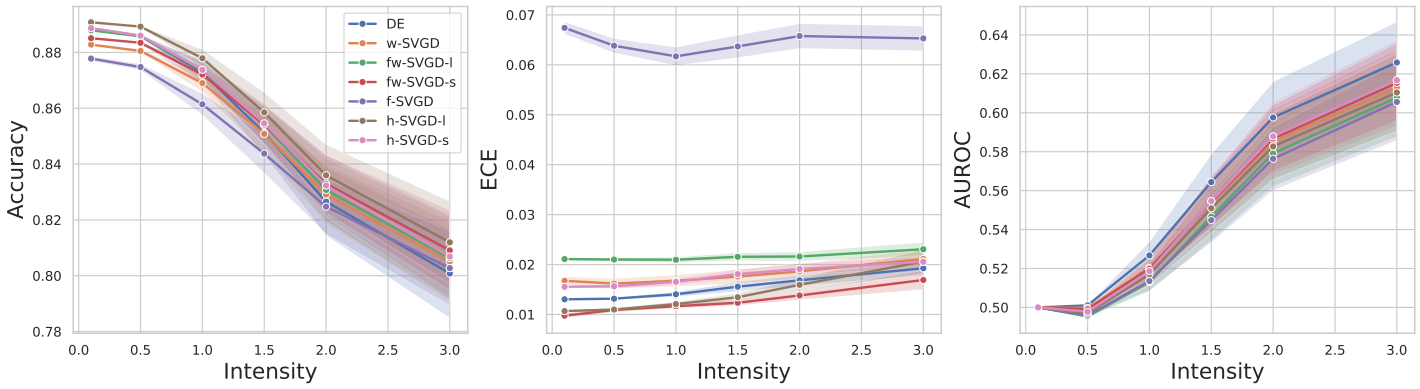


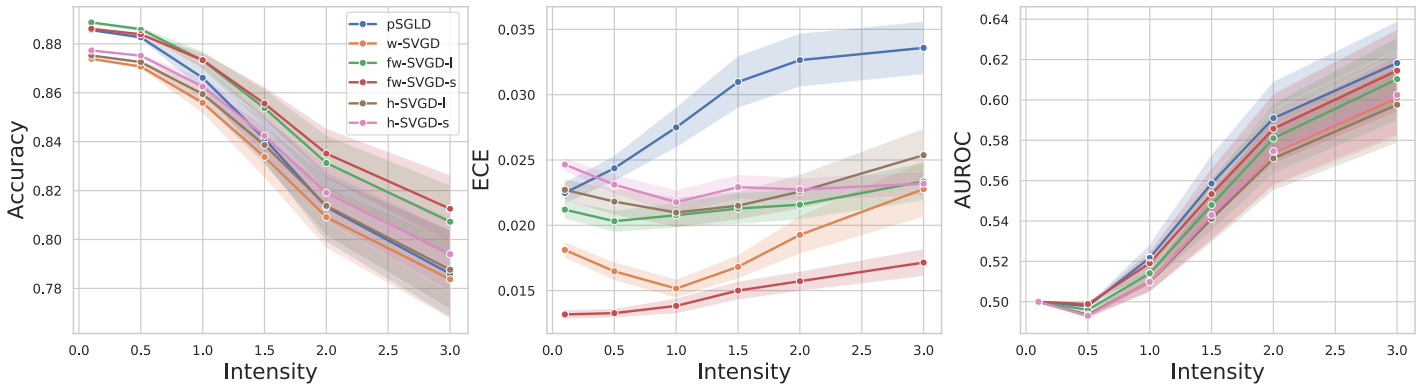

*Figure B.3.* **Calibration curves CIFAR-10:** calibration curves showing the relationship between uncertainty (horizontal axis) and accuracy (vertical axis) for the stochastic methods $\beta = 1$ (right) and deterministic $\beta = 0$(left).

### B.3. Corrupted data

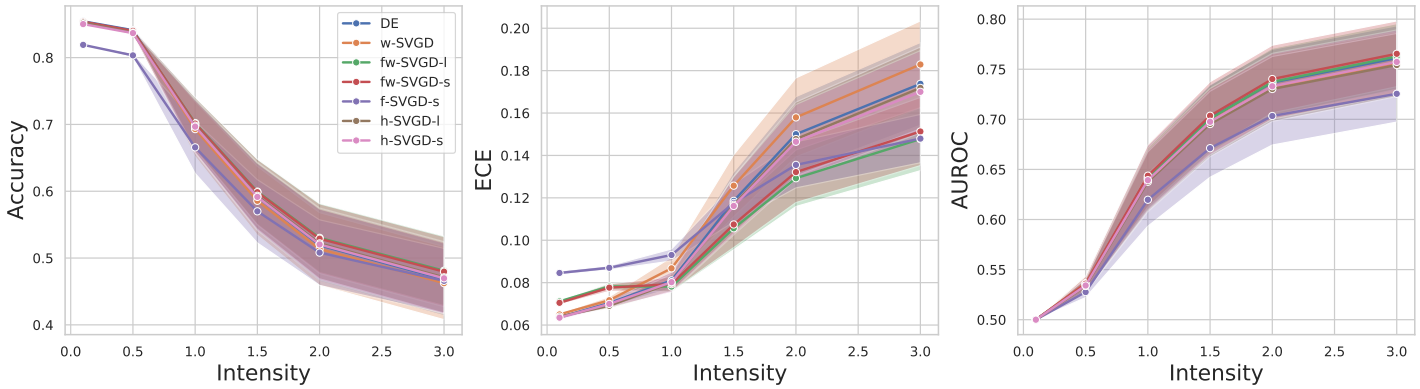
We studied the accuracy, ECE, and AUROC score of the different methods illustrated in the main text on datapoints corrupted with different intensities. The corruption is created by blurring the images in the test set with random Gaussian blur. The intensity of the corruption is determined by the standard deviation of the Gaussian kernel used for the blurring, which is selected from $\sigma \in \{0.1, 0.5, 1, 1.5, 2.0, 3.0\}$. The kernel size is fixed to 7. The results for FashionMNIST are reported in Figure B.4a for the deterministic case and Figure B.4b for the stochastic one. The first one shows that the Deep Ensemble has a tendency of considering the corrupted data OOD; for that reason that accuracy is decreasing faster and the AUROC is the most elevated, this fact could also be interpreted as a tendency of deep ensemble to overfit. On the other side the functional methods that we introduced seem more robust in the sense that the accuracy on the corrupted data remains high, especially for the h-SVGD-l and for the same reason the AUROC is lower. Interestingly, the f-SVGD leads to less calibrated predictions when compared to all other methods. For the stochastic variants, instead, the fw-SVGD-s and fw-SVGD-l seem to be the more robust in accuracy and the pSGLD displays characteristics very similar to the Deep Ensemble. The results for CIFAR-10 in Figure B.4c and Figure B.4d show a similar trend for all methods in terms of accuracy and AUROC, the only relevant difference is in the ECE for which the fw-SVGD appears more calibrated across the corruptions for the deterministic case and the h-SVGD-l for the stochastic one. The worse performance of f-SVGD for bigger models is confirmed also in this experiment.



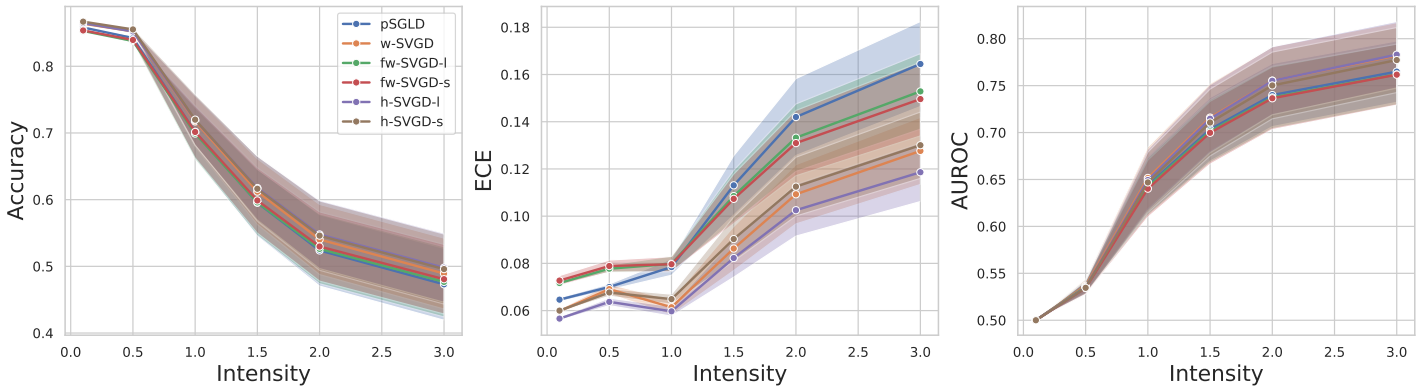

(a) **Corruption analysis FashionMNIST** $\beta = 0$**.**




(b) **Corruption analysis FashionMNIST** $\beta = 1$**.**




(c) **Corruption analysis CIFAR-10** $\beta = 0$**.**




(d) **Corruption analysis CIFAR-10** $\beta = 1$**.**

*Figure B.4.* we compare accuracy, ECE, and AUROC under corruption of the FashionMNIST and CIFAR10 dataset, given by Gaussian blur. For each method, we show the mean and standard error on the corrupted test set for different intensities of corruption.

## C. Related Work

We have already addressed some of the related work in Section 2 and additionally highlight previous relevant work on ensemble-based inference methods for deep neural networks here.

**Particle-based inference.** Particle-based methods for flexible inference have gained a lot of momentum since the publication of the original SVGD method (w-SVGD) (Liu & Wang, 2016). Out of these, standard SVGD is still the best studied method, with many theoretical convergence results and strong guarantees (Liu, 2017; Liu et al., 2019; Korba et al., 2020; Duncan et al., 2019; Chewi et al., 2020). Interestingly, it has also been shown that SVGD performance can benefit from projecting the particles into a different space (Chen & Ghattas, 2020), which is very similar to our projection from weight space into function space. Moreover, our approximation of the function space Stein discrepancy using mini-batches is related to the idea of stochastic Stein discrepancies, which have been shown to offer good approximations in practice (Gorham et al., 2020). Apart from that, many extensions of SVGD have been proposed, including using second-order methods (Detommaso et al., 2018), additional Langevin noise (Gallego & Insua, 2018), importance-weighting (Yao et al., 2020), non-Markovian trajectories (Ye et al., 2020), explicit noise models (Chang et al., 2020), and matrix-valued kernels (Wang et al., 2019a). While we study standard SVGD in this work, most of these extensions could be readily applied in our framework, which will be an exciting avenue for future work.

**BNN inference.** Bayesian neural networks have a long history, dating back to MacKay (1992) and Neal (1995). While Markov Chain Monte Carlo (MCMC) inference is considered the gold standard for these models (Neal, 1995; Wenzel et al., 2020a; Fortuin et al., 2021b; Garriga-Alonso & Fortuin, 2021; Fortuin et al., 2021a; Fortuin, 2021; Izmailov et al., 2021), it is often costly and in practice replaced by variational inference (VI) approaches (Blundell et al., 2015; Swiatkowski et al., 2020; Dusenberry et al., 2020; Immer et al., 2021). So far, SVGD has been applied to BNNs only sparingly (Hu et al., 2019; Wang et al., 2019b). These approaches correspond to special instantiations of our framework. Moreover, many practical techniques for improving BNN performance can also be applied in our framework: posterior tempering (Mandt et al., 2016; Wenzel et al., 2020a) corresponds to the temperature factor in the posterior, and the cyclical annealing schedules from VI (Fu et al., 2019) and MCMC inference (Zhang et al., 2019) are very related to the annealed SVGD (D'Angelo & Fortuin, 2021).

## D. Implementation details

In this section, we report details on our implementations in the experiments we performed. All the experiments were performed on an internal cluster with NVIDIA GTX 1080 Ti and took roughly 200 GPU hours.

### D.1. 1D regression

We generate 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 10000 gradient steps, a learning rate of 0.001 and batchsize 64. The kernel bandwidth is estimated using the median heuristic. We test the models on 100 uniformly distributed points in the interval $[0, 7]$. The random seed was fixed to 42.

### D.2. 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 5000 gradient steps, a learning rate of 0.01 and batchsize 64. The kernel bandwidth is estimated using the median heuristic. We use the hyperbolic annealing in the first 2000 iterations. The random seed was fixed to 42.

### D.3. Classification on FashionMNIST

On this dataset, we used a feed-forward neural network with 3 hidden layers and 100 hidden units with ReLU activation function.The likelihood was softmax and the prior standard normal $\mathcal{N}(0, \mathbb{I})$. We used 50 particles initialized with random

samples from the prior and optimize them using Adam (Kingma & Ba, 2014) for 60000 steps. For $\beta = 0$ the learning rate was 0.0025 for h-SVGD-l, h-SVGD-s, fw-SVGD-s,fw-SVGD-l,DE and 0.005 for f-SVGD, w-SVGD, we used the hyperbolic annealing (D'Angelo & Fortuin, 2021) for h-SVGD and fw-SVGD and the linear annealing for f-SVGD in the first 1000 steps. For $\beta = 1$ instead we used a learning rate of $0.0025$ for fw-SVGD-l,pSGLD, and 0.005 for fw-SVGD-s,h-SVGD-s,h-SVGD-l and w-SVGD, the hyperbolic annealing (D'Angelo & Fortuin, 2021) was applied to h-SVGD and fw-SVGD for the first 1000 steps. A batchsize of 256 was used for all experiments. The kernel bandwidth is estimated using the median heuristic for all different methods. The learning rates were searched over the following values $(1e-4, 5e-4, 1e-3, 25e-4, 5e-3)$. We tested for 60000 and 30000 total number of steps and batchsize 256 and 128. All methods with a repulsive force were tested using hyperbolic, cyclical, linear and without annealing. All results in Table 1 are averaged over the following random seeds $(38, 39, 40, 41, 42)$.

### D.4. 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 used 20 particles initialized using He initialization (He et al., 2015) and optimized them using Adam (Kingma & Ba, 2014) for 35000 steps. For $\beta = 0$ we use a lerning rate of $0.001$ for all methods apart for f-SVGD were we used $0.0001$ given that a bigger value was always leading to numerical instabilities. For $\beta = 1$ the learning rate was fixed to 0.001 for all methods, we used linear annealing only for h-SVGD-l, h-SVGD-s for the first 1000 steps. The batchsize was 128 for all experiments. The kernel bandwidth is estimated using the median heuristic for all different methods. The learning rates were searched over the following values $(1e-4, 1e-3, 5e-3)$ we tested for 20 and 10 particles and all methods with a repulsive force were tested using hyperbolic, cyclical, linear and without annealing. All results in Table 1 are averaged over the following random seeds $(38, 39, 40, 41, 42)$.