Efficient Uncertainty Estimation via Sensitivity-Guided Subnetwork Selection for Scalable Variational Inference

Anonymous authors
Paper under double-blind review

Abstract

Quantifying predictive uncertainty with minimal computational overhead remains a significant challenge for reliable deep learning applications in safety-critical systems. While Bayesian neural networks (BNNs) are the gold standard for uncertainty quantification, they require considerable training time and computational resources. Although a body of work has focused on mitigating the computational cost of BNN inference via post-hoc approaches, efforts to accelerate training and convergence remain limited. This paper proposes a partial Bayesian training approach via mean-field variational inference (VI), enabling controllable uncertainty modeling through sparse gradient representations. The selection of the variational Bayesian subnetwork is guided by a first-order gradient sensitivity analysis, which is grounded in uncertainty propagation theory. Under mean-field assumptions, we demonstrate how this framework effectively informs the selection of parameters that represent the network's predictive uncertainty. This criterion is also efficiently integrated into auto-differentiation tools avoiding additional computational burdens. The resulting model consists of a combination of deterministic and Bayesian parameters, facilitating an effective, yet efficient, representation of uncertainty. We investigate the effects of varying the proportion of Bayesian parameters (ranging from 1% to 95%) across diverse tasks, including regression, classification, and semantic segmentation. Experimental results in MNIST, CIFAR-10, ImageNet, and Cityscapes demonstrate that our approach achieves competitive performance and uncertainty estimates compared to ensemble methods. While maintaining substantially fewer parameters, approximately 50%, 80% less than full VI and ensembles, our approach offers reduced training costs with faster convergence compared to full or partial VI trained from scratch. Furthermore, we assess the robustness of predictive uncertainty in the presence of covariate shifts and out-of-distribution data, demonstrating that our method effectively captures uncertainty and exhibits robustness to image corruptions.

1 Introduction

Robust uncertainty quantification is essential to ensure reliable decision-making in safety critical systems Nguyen et al. (2015); Guo et al. (2017); Yang et al. (2023); Cao et al. (2024), such as medical diagnosis Rajpurkar et al. (2022) and self-driving cars Bojarski et al. (2016). Predictive uncertainty is a quantitative metric, providing insights into the model's confidence in its predictions and facilitating risk assessment for real-world deployment. Risk assessment related to distributional shifts in real-world environments compared to the training i.i.d. dataset is enabled through calibrated predictive uncertainty Malinin et al. (2021; 2022); Ovadia et al. (2019). Poorly calibrated probabilities are associated with silent failures, where models exhibit overconfident false predictions Guo et al. (2017). Bayesian neural networks (BNN) Ghahramani (2015) provide an approach to learning model uncertainty over parameter distributions; BNNs suffer from a large parameter space, slow convergence, and noisy loss landscapes limited by sampling Jospin et al. (2022). As a result, applications of BNNs to complex tasks such as large multi-class classification or segmentation remain challenging Ovadia et al. (2019). Deterministic network pruning, predefined sparsity, and learning sparse networks have shown success in preserving predictive performance while reducing computational overhead Evci et al. (2020); Frankle & Carbin (2019); Mozer & Smolensky (1988); LeCun et al. (1989).

Partial Bayesian networks are effective at preserving predictive uncertainty over full-BNNs; however, current efforts rely on an iterative selection of the layer-wise subnetwork Zeng et al. (2018); Sharma et al. (2023), constructing a low dimensional parameter space through PCA Izmailov et al. (2020), or on a computationally limiting Hessian-based subnetwork selection Daxberger et al. (2021). We propose a computationally low-cost method based on first-order gradients to select a sparse subnetwork for partial Bayesian inference to reduce overparameterization in BNNs; our contributions are as follows:

1. Introduce an algorithm for training scalable partial Bayesian networks (PBN)[1] that selectively assigns a small portion of Bayesian parameters based on first-order gradient analysis;
2. Conduct extensive empirical assessments on large scale classification, and semantic segmentation tasks. We show a competitive performance while reducing the required Bayesian parameters by $> 95\%$ compared to a full variational inference approach and 80% compared to ensembles;
3. Evaluate our predictive uncertainty and its reliability on out-of-distribution testing on covariate data shifts and unseen classes.

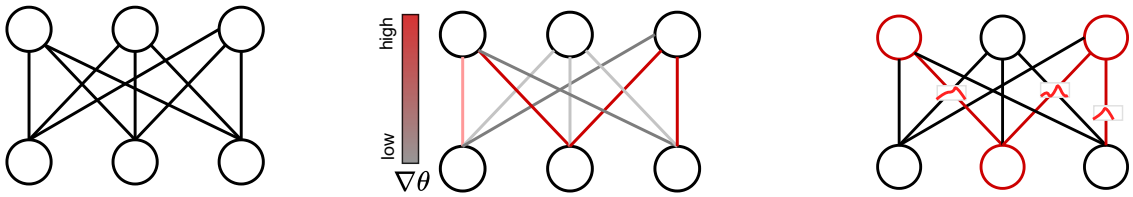
2 Related works

Uncertainty Estimation. Predictive uncertainty can be quantified through various deep neural network (DNN) architectures. Bayesian neural networks (BNN) learn parameter distributions as opposed to point estimates, allowing estimation of the epistemic uncertainty (i.e. the model uncertainty) Abdar et al. (2021). BNNs can be approximated via variational inference (VI) Blundell et al. (2015), Markov Chain Monte Carlo (MCMC) Welling & Teh (2011), dropout Monte-Carlo samples Gal & Ghahramani (2016); Kingma et al. (2015), and expectation propagation Hernández-Lobato & Adams (2015). A popular non-Bayesian predictive uncertainty method is model-ensembling Lakshminarayanan et al. (2017), where multiple models are trained deterministically, and their predictions are combined to estimate predictive uncertainty. Despite the plethora of predictive uncertainty quantification methods Abdar et al. (2021), most exhibit high computational costs at training and inference times.

Overparameterization. Reducing the parameter complexity has proven to be a successful strategy when training sparse deterministic neural networks Guo et al. (2016); Dey et al. (2019); Evci et al. (2020); LeCun et al. (1989). For instance, achieving sparsity in deterministic models has been realized through connection pruning Guo et al. (2016); LeCun et al. (1989), random pre-defined sparsity Dey et al. (2019), or training sparse models with a dynamic drop-and-grow algorithm based on parameter saliency Evci et al. (2020). Parameter saliency is frequently used to guide the selection of important parameters or to determine pruning criteria Yeung et al. (2010). Parameter importance is assessed through sensitivity analysis, which examines how perturbations in a parameter affect the output of the objective (or loss) function. Due to computational constraints, sensitivity analysis typically relies on first- and second-order gradients due to the computational burden associated with higher-order gradient computations Yeung et al. (2010). First- and second-order based parameter sensitivity methods have been used to selectively prune neural network parameters without impacting model performance as in-training Evci et al. (2020); Shi et al. (2020), post-hoc methods Mozer & Smolensky (1988); Molchanov et al. (2019), and iterative pruning Frankle & Carbin (2019). These techniques have primarily been applied within deterministic neural networks; we now discuss parameter reduction strategies in the context of probabilistic (Bayesian) models.

Computational Demand of BNNs. Multiple techniques have successfully sped up inference times of BNNs Jia et al. (2021); Subedar et al. (2021) as a post-hoc processing method. However, this approach is limited as it relies on training a dense BNN as a first step and is followed by a post-hoc analysis for pruning Sharma & Jennings (2021) or quantization of the network Subedar et al. (2021); Ferianc et al. (2021). Such post-hoc solutions lead to additional computational costs. Another limitation is that BNNs are prone to convergence issues and noisy loss landscapes driven by limited samples drawn from the posterior distribution Jospin et al. (2022); Ovadia et al. (2019). In theory, increasing the number of samples drawn during training

[1]Partial Bayesian networks in this context refers to sparsity introduced into the sigma parameter, rendering the network "partially" Bayesian.



(a) Deterministic Training (b) Subnetwork Selection (c) Partial VI Training

Figure 1: Our proposed method for training partial variational Bayesian networks (PBNs) consists of three stages: (a) Train a deterministic model by minimizing the negative log-likelihood, with parameters θ initialized randomly. (b) Perform first-order gradient-based sensitivity analysis to compute $\nabla\theta$ and identify the top-k parameters by magnitude $|\nabla\theta|$, which are then modeled as stochastic variables; the rest remain deterministic. The learned weights from (a) are used to initialize this step. (c) Train the resulting PBN using variational inference by maximizing the Evidence Lower Bound (ELBO).

leads to a more accurate approximation of the posterior weight distributions. However, the practicality of this approach is often constrained by the computational overhead associated with drawing a large number of samples for each gradient update step Jospin et al. (2022).

Partial Bayesian Networks. Partial Bayesian learning leverages the computational efficiency of deterministic networks with the probabilistic representation of Bayesian parameters. Zeng et al. (2018) investigated the placement of Bayesian layers in active learning, showing that using one or two near the output outperforms fully Bayesian models. A more recent study has explored partial Bayesian learning to estimate uncertainty within a brain tumor segmentation model Prabhudesai et al. (2023). Daxberger et al. (2021) utilizes a linearized Laplace approximation MacKay (1992) to infer a Bayesian subnetwork within a pre-trained deterministic network. At inference, they employ an approximate Hessian with a generalized Gauss-Newton (GGN)-Laplace method to infer a full-covariance Gaussian posterior. Their method is validated on regression and classification tasks on MNIST and CIFAR10 datasets. While achieving competitive uncertainty quantification akin to ensembles, their approach requires the computation and storage of the full covariance matrix, rendering it intractable for tasks such as semantic segmentation or large-multi-class classification.

Sharma et al. (2023) evaluated stochastic subset inference on a two-step training approach as Daxberger et al. (2021) through Hamiltonian Monte Carlo (HMC) Neal (2012), Laplace Approximation, SWAG Maddox et al. (2019) on regression and one step joint training using stochastic variational inference (SVI) Blundell et al. (2015) on classification. They analyzed the impact of sub-stochastic parameter placement concluding that stochastic input layers yield the highest accuracy, while stochastic last-ResBlock and output layers have the lowest negative log-likelihood. More recently, a variational Bayesian last layer approach offering sampling-free VI Harrison et al. (2024) proposes a more deterministic formulation for training Bayesian last layers for addressing the sampling issue associated with BNNs, however the method introduces additional hyperparameters that potentially limit its practical introduction for real-world applications.

Our contribution builds on prior work in subnetwork inference Daxberger et al. (2021); Abboud et al. (2024) by introducing a computationally efficient, gradient-based sparsity criterion within variational Bayesian neural networks. This approach reduces parameter complexity, accelerates convergence, and enables scalable mean-field variational inference on large datasets such as ImageNet Deng et al. (2009).

3 Method

Given a dataset $\mathcal{D}$, with input samples x_i and targets y_i, we aim to train a neural network $f_\Theta(\cdot)$, parameterized by $\Theta = \theta_1, \theta_2, ..., \theta_N$ for a given task by minimizing the error E, such that:

$$\min_{\Theta} E(\mathcal{D}, \Theta) = \min_{\Theta} E(y|x; \Theta) \tag{1}$$

To determine the uncertainty in the output y of the neural network $f_\Theta(\cdot)$, we must first account for the uncertainties arising from its constituent parameters. This can be achieved by leveraging the theory of error propagation Ku et al. (1966); Zurada et al. (1994). Given a network f, parameterized by Θ, the output y varies with Θ such that $f_\Theta(x) = y$ for a fixed input data point x. Our goal is to establish the relationship between small perturbations in the network's parameters $\Delta\Theta$ and the output y. This relationship can be expressed as the derivative of f with respect to Θ at a given data point x_i, i.e., $\frac{\partial f(x_i)}{\partial \Theta} = \frac{\partial y}{\partial \Theta}$. The uncertainty in the output is directly proportional to this slope, and it can be propagated using the Taylor series expansion of the variance of y:

$$(\Delta y)^2 \approx \left(\frac{\partial y}{\partial \Theta}\right)^2 (\Delta\Theta)^2 \tag{2}$$

where $(\Delta y)^2$ and $(\Delta\Theta)^2$ represent the variances in y and Θ, respectively. Since we are employing mean-field theory, we can decompose the right-hand side into the individual network parameters, leveraging the mean-field assumption of parameter independence Blei et al. (2017):

$$(\Delta y)^2 \approx \left(\frac{\partial y}{\partial \theta_1}\right)^2 (\Delta\theta_1)^2 + \left(\frac{\partial y}{\partial \theta_2}\right)^2 (\Delta\theta_2)^2 + \ldots + \left(\frac{\partial y}{\partial \theta_N}\right)^2 (\Delta\theta_N)^2 \tag{3}$$

$$\therefore (\Delta y)^2 = \sum_{i=1}^{N} \left(\frac{\partial y}{\partial \theta_i}\right)^2 (\Delta\theta_i)^2 \tag{4}$$

In equation 4, $(\Delta y)^2$ and $(\Delta\theta_i)^2$ represent the uncertainties in the output and the individual network parameters, respectively. The term $\frac{\partial y}{\partial \theta_i}$ quantifies the sensitivity of the output uncertainty to the uncertainty in each parameter θ_i. Therefore, leveraging sensitivity analysis, we can identify a subset of network parameters to which the output uncertainty exhibits the highest sensitivity. These parameters can then be trained probabilistically, allowing us to capture the uncertainty of the network with a reduced set of parameters, thereby improving efficiency while maintaining the ability to model the network's predictive uncertainty.

To implement this, we propose a method for training Partial Bayesian Networks, as depicted in Figure 1 and Algorithm 1. Given a pre-trained deterministic model, a partial Bayesian model is then trained with MFVI with a Gaussian prior, initializing the mean values μ with the point estimates from the deterministic model. The selection of Bayesian parameters is based on first-order gradient sensitivity analysis, where the number of Bayesian parameters is controlled by the hyperparameter r_{bayes}, which adjusts the proportion of Bayesian parameters in the final Partial Bayesian Network.

Algorithm 1 Partial Bayes with Sparse Gradients

Training Step 1: Deterministic
  Input: Dataset D. Initialize Θ_d. **Learn** f_{Θ_d} by minimizing $\mathcal{L}(f_\theta(x_i), y_i)$. **Output**: f_{Θ_d}
Training Step 2: Partial Bayesian
  Input: Network f_{Θ_d}, dataset D, Bayesian rate r_{bayes}, Posterior init parameters $\mu_{post}, \sigma_{post}$
  Initialize:
    $\Theta_b; (\mu_b, \sigma_b) \leftarrow$ Bayes parameters
    $\nabla_\Theta \mathcal{L} \leftarrow$ compute gradients
    $k = \text{Top}k(|\nabla_\Theta|, k) \leftarrow$ Sensitivity Analysis $(k = r_{bayes} \times N_{\theta_{total}})$
    $\mu_{b,d} = \theta_d \leftarrow \Theta_d, \quad \sigma_b = log(1 + exp(\rho_b)), \quad \rho_b \sim \mathcal{N}(\mu_{post}, \sigma_{post}), \quad \sigma_d = 0$
  Learn $f(\Theta_b, \Theta_d)$ by minimizing $\mathcal{L}(f_\theta(x_i), y_i) + \beta \cdot KL(q(\theta_b), p(\theta_b))$

Notation: We aim to learn a partial Bayesian neural network $f_{\Theta_b}(\cdot)$, where Θ_b is parameterized by $(\mu_b, \sigma_b) \in \mathbb{R}^N$, and their respective means μ_b are initialized from a pre-trained deterministic neural network f_{Θ_d}, where $\Theta_d \in \mathbb{R}^N$. For weights designated as Bayesian, their corresponding sigma values are modeled as the softplus ($\sigma = log(1 + exp(\rho))$) of a randomly initialized ρ parameter sampled from a Gaussian distribution $\mathcal{N}(\mu_{post}, \sigma_{post})$ and minimized by the KL divergence against a standard normal prior $\mathcal{N}_{prior}(0, I)$. The loss is defined by the evidence lower bound criterion (ELBO), $\mathcal{L}_{ELBO} = \sum_i \mathcal{L}(f_{\Theta_d}(x_i), y_i) + \beta \cdot KL(q(\theta_b), p(\theta_b))$, where $q(\theta_b)$, $p(\theta_b)$ are the posterior and prior terms over the parameters, β is the weight of the KL divergence term. The KL weight term, β, is gradually annealed over training epochs, from an initial value β_{init} to a target value β_{target} according to the schedule $\beta_i = \beta_{target} - (\beta_{target} - \beta_{init}) \times (epoch_i / epoch_{total})$. Subscripts b and d represent Bayesian and deterministic parameters, respectively.

Training the Deterministic Network: The first step in our approach is to train a deterministic model on a given dataset $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$ composed of i.i.d. data points. We train a neural network to model the function $f_{\Theta_d}(\cdot)$ parameterized by Θ_d, by minimizing the negative log-likelihood $\mathcal{L}(f_{\Theta_d}(x_i), y_i)$ as a standard neural network with point estimates.

Initialization of Partial Bayesian Network: Before training the partial Bayesian network, a gradient-based sensitivity analysis Yeung et al. (2010) (Algorithm 1- Step 8) is used to select the weights with the highest magnitude gradients, noted as Top$k(|\nabla_\Theta|, k)$, where k is the number of parameters to set as Bayesian based on an input hyperparameter r_{bayes} which allows us to adjust the probabilistic extent of the network. We use first-order gradients to select the parameters with the highest-magnitude gradients and reparameterize them as variational parameters. The benefit of using first-order approximations is that it is provided by automatic differentiation tools such as PyTorch at no additional computational cost to the workflow Paszke et al. (2017).

The Top$k$ first-order gradient selection method allows us to have mixed deterministic and Bayesian parameters within individual layers or filters, where, in practice, sparsity is introduced into the σ parameter. Deterministic parameters are modeled as $\mathcal{N}_i(\mu_i, 0) = \delta_i(\mu_i)$ delta functions, while Bayesian parameters are modeled as $\mathcal{N}_i(\mu_i, \sigma_i)$ distributions. The μ_i parameter values are initialized using the point estimates derived from the deterministic model. This approach ensures a robust starting point for optimizing uncertainty learning in the partial Bayesian network.

For every layer l in the model, a mask is generated using the gradient values of its weights. The masks set the initial values for the σ parameter linked to each Bayesian weight and form the sparse gradient updates. $\rho^{[l]}$ represents the ρ parameter for the l^{th} layer in the model the associated mask is used to initialize the ρ_b values of the Bayesian weights to a random value drawn from $\mathcal{N}(\mu_{post_i}, \sigma_{post_i})$ and ρ_d values of the deterministic weights to be set to a near-zero value for numerical stability. The deterministic ρ_d values are not updated in the computational graph; their gradients are set to zero and do not contribute to the KL Divergence term. In effect, deterministic weights are modeled as $\delta_{ij}(\mu_{ij})$, while Bayesian weights are modeled as $\mathcal{N}_{ij}(\mu_{ij}, \sigma_{ij})$. The μ_{ij} parameter values are initialized using the point estimates derived from the pre-trained model. This approach ensures a robust starting point for optimizing uncertainty learning in the partial Bayesian network.

Training the Partial Bayesian Network: The partial Bayesian network (PBN)[2] is trained using variational inference Kingma & Welling (2014) by employing the reparameterization trick Kingma et al. (2015). Five posterior samples were chosen, as experimenting with more samples slightly improved performance but significantly increased training time. Comparative 5-member ensembles were run as a comparative baseline. The network is trained to minimize the -ELBO loss described in Algorithm 1 and Notation subsection above. To ensure that the standard deviation σ remains strictly positive, we reparameterize it using a latent variable ρ parameter, where $\sigma = log(1 + exp(\rho))$ i.e., the softplus transformation of ρ. During this step, both deterministic and Bayesian parameters are jointly updated.

Sparse Gradient Updates: The layers of the PBN consist of a combination of deterministic and Bayesian parameters requiring customization of the forward and backward passes. Sparse gradient representations enable partial updates to the ρ_i parameters. The gradient tensor associated with the ρ tensor is then converted into a sparse representation, where $\nabla\rho_d = 0$; $\forall\theta_d$ instructing the optimizer to disregard these

[2]partial Bayesian network (PBN) with $r_{\text{bayes}} = R\%$ is referred to in the paper as PBN R% or Partial R%

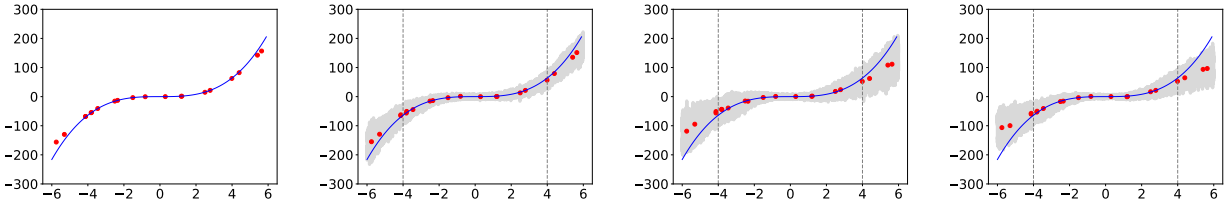
elements within the computational graph. Consequently, the ρ_d elements associated with deterministic weights remain unchanged in the computational graph, as their gradients are zero. This is in contrast with the methods described by Prabhudesai et al. (2023); Zeng et al. (2018) that require calculating and storing full gradients for each ρ parameter at each optimization step.

4 Experimental Results

The following experiments are designed to demonstrate the versatility of the proposed method across a range of tasks, from simple regression to multi-class classification and complex multi-class segmentation, with network sizes spanning from 10^2 to 10^7 parameters. It is important to note that the primary focus of these experiments is not on achieving state-of-the-art performance; rather, the objective is to investigate the convergence behavior of Partial Bayesian Networks (PBN) under varying levels of model stochasticity (r_{bayes}). Here, convergence refers to the minimization of the loss function to reach acceptable model performance. Specifically, we explore whether models can converge to solutions that retain deterministic task-specific performance, effectively capture uncertainty, and minimize the total number of parameters required. To evaluate the robustness of the models, we also conduct stress tests using out-of-distribution datasets and covariate distributional shifts. The key questions addressed in these experiments include: 1) What is the distribution of the selected subnetwork parameters within the network, and how does it compare with previous work? 2) Does initializing the PBN's mean (μ) parameter from deterministic point estimates facilitate faster convergence? By systematically exploring these questions, we aim to provide deeper insights into the behavior and performance of partial Bayesian models across different tasks and initialization strategies.

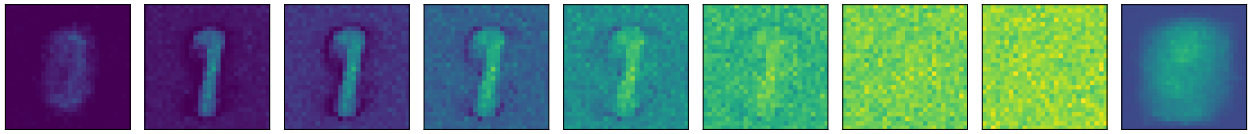
Datasets: We evaluated our methods on several benchmark datasets. For classification, we used MNIST LeCun (1998), CIFAR-10 Krizhevsky et al. (2009), and ImageNet Deng et al. (2009). The segmentation experiments were conducted on the Cityscapes dataset Cordts et al. (2016). To assess robustness under covariate shift, we used the CIFAR-10-C and ImageNet-C datasets Hendrycks & Dietterich (2019) for out-of-distribution (OOD) testing.

Evaluation metrics: Accuracy and intersection-over-union (IoU) is used to evaluate classification and segmentation performances, respectively. The Brier score is used as a proper scoring rule for measuring uncertainty Brier (1950); Ovadia et al. (2019) for Brier Score $= \frac{1}{N}\sum_{i=1}^{N}(\delta_{i=y} - p_\theta(y = C \mid x))^2$. Negative log-likelihood (NLL) is also used to evaluate the quality of uncertainty Ovadia et al. (2019). Entropy of Expectation (EoE) is used to compute the total uncertainty EoE $= -p\left(y = c \mid x\right)\log\left(p\left(y = c \mid x\right)\right)$. Floating point operations (FLOPs) and the number of model parameters are used to measure computational efficiency (details in Appendix:C).



(a) Deterministic (b) $r_{bayes} = 20\%$ (c) $r_{bayes} = 80\%$ (d) VI

Figure 2: Results on $y = x^3 + \epsilon$ toy dataset. Predictions on toy data, the blue line is ground truth $y = x^3$, red dots are sample test predictions, and the gray shaded area is the 3σ confidence interval. The vertical lines highlight the in- and out-of-distribution (ID, OOD) test points. The proposed model displays how varying r_{bayes} allows the model to capture the predictive uncertainty for ID and OOD samples. A figure with a range of r_{bayes} values is in Appendix: F.



(a) 5% (b) 10% (c) 20% (d) 40% (e) 60% (f) 80% (g) 95% (h) VI (i) Ens.

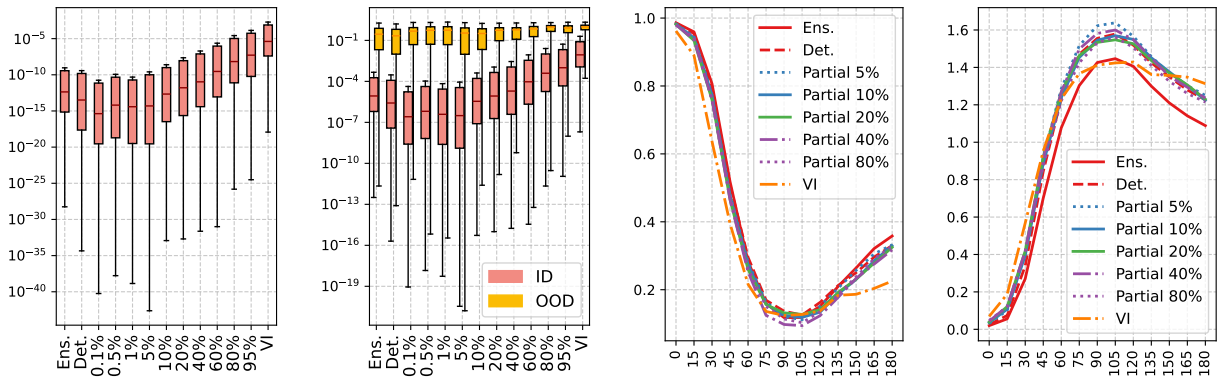
Figure 3: MNIST Weight uncertainty (σ): heatmaps indicate the value of the weight uncertainty and highlight the location of the (selected) Bayesian weights in PBN as a function of r_{bayes} (a-g), fully VI (h), and 5-member ensemble (i). The images highlight how partial Bayes focuses on the uncertainty of the important features within the given dataset.

4.1 Regression on Toy Dataset

We first evaluate our approach qualitatively on a one-dimensional regression dataset. We follow the same experimental setup described by Hernández-Lobato & Adams (2015). The toy dataset is drawn from $y = x^3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 9)$. 20 samples are uniformly drawn from a training interval $[-4, 4]$. 30,000 training examples was used for training a multi-layer perception (MLP) with a single hidden layer of 100 units. The model was tested with 10,000 samples drawn uniformly from $[-6, 6]$. Deterministic, VI, and partial Bayesian models with various r_{bayes} were trained. Figure 2 illustrates how the partial Bayesian approach is capable of capturing the uncertainty with less than half of the parameters set as Bayesian.

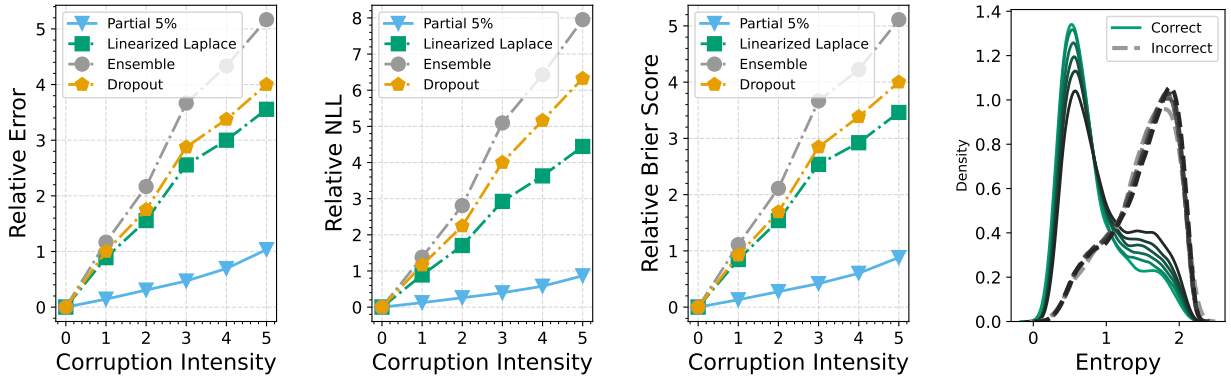
4.2 MNIST Demo

For a demonstrative image classification, we use a LeNet LeCun et al. (1998) network to evaluate performance on MNIST LeCun (1998) classification. Figure 4a shows performance trends of different models as a function of r_{bayes}. An increase in the Bayesian rate (r_{bayes}) with a fixed number of training steps results in a decline in test performance, particularly in model uncertainty. Figure 3 illustrates the evolution of weight uncertainty (σ parameter) in the network's initial layer as a function of r_{bayes}. At $r_{bayes} = 5\%$, Bayesian weights focus on central connections where the dominant MNIST features are located. With increasing r_{bayes} values approaching the VI model, weight uncertainty increases 4-fold, introducing randomness into weight uncertainty representation. This loss may be due to insufficient optimization steps for network convergence (Figure 3 and Figure 11). From an out-of-distribution (OOD) performance, we compare the performance of the models with rotated versions of MNIST data simulating a covariate shift and also test the performance



(a) Brier Score on ID (b) Entropy ID vs. OOD (c) Accuracy vs Rot. (°) (d) Brier Score vs. Rot. (°)

Figure 4: Results on MNIST: (a) Brier score for test set with increasing stochasticity. (b) Entropy for MNIST (ID: in-distribution) fashionMNIST (OOD: out-of-distribution). (c) Accuracy and (d) Brier score on Rotated MNIST with angle in degrees on the x-axis. (Ens. = Ensemble, Det. = Deterministic, VI=Variational Inference, Rot. (°)= Rotation in degrees)



(a) ΔError ↓ (b) ΔNLL ↓ (c) ΔBrier Score ↓ (d) Entropy

Figure 5: Calibration under covariate distributional shift of CIFAR10 with models: (ours) Partial with $r_{bayes} = 5\%$, Linearized Laplace Daxberger et al. (2021), ensemble, dropout. The x-axis is the corruption intensity of the CIFAR10 for 19 different corruptions in CIFAR10-C, and the y-axis is (a) relative error rate, (b) relative NLL, (c) relative Brier Score. The relative measure is with respect to the models' performance on the uncorrupted data. (d) Entropy for correctly and incorrectly classified examples for both ID and OOD samples for proposed method (Partial VI) with r_{bayes}=5% . Linearized Laplace, ensemble, and dropout results are from Daxberger et al. (2021).

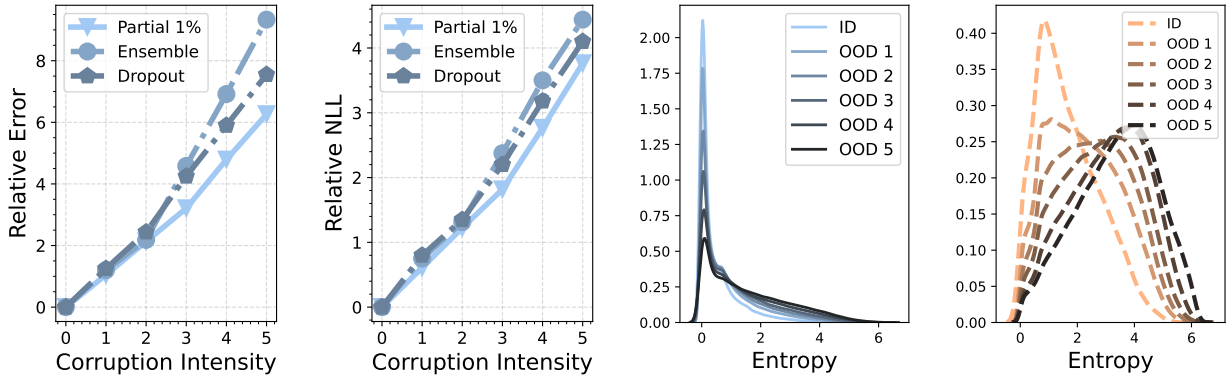
on a different dataset, where the model is trained on MNIST and tested on FashionMNIST, the entropy of the in-distribution (ID) and OOD is shown in Figure 4. With respect to covariate shift (rotated MNIST), ensembles display more robustness to data shift consistent with Ovadia et al. (2019) findings. In contrast, when evaluated on entirely different OOD data, the PBN approach demonstrates superior performance, with a larger entropy gap between ID and OOD samples, particularly for models with $r_{\text{bayes}} \leq 5\%$, indicating enhanced OOD detection capabilities.

4.3 Large Scale Networks

To assess whether our approach holds on large-scale models such as ResNet18He et al. (2016). As in previous sections, we demonstrate that very low percentages of Bayesian parameters exhibit competitive performance; therefore, in this section, we evaluate our method with r_{bayes} ≤5%. We train and evaluate ResNet18 on CIFAR10 with $r_{bayes} = 5\%$ on test data, and on corrupted data from CIFAR10-C Hendrycks & Dietterich (2019). Figure 5 shows our method's robustness against corruption, where relative to the model's performance on uncorrupted data, the model's error rate, Brier score, and NLL are less impacted than the performance of subnetwork inference by linearized Laplace Daxberger et al. (2021). The robustness of our partial Bayesian networks trained with variational inference (VI) is consistent with conclusions from the uncertainty study by Ovadia et al. (2019), where models converge to a lower accuracy but are much more robust to data shifts.

Our approach relies on pre-trained point estimates for the selection of the Bayesian parameters, but does initializing the PBN model with deterministic weights also help speed up the training of PBN? To answer this question, we compare the training FLOPs of a full VI network to a partial Bayesian network with 5% Bayesian parameters. For the partial Bayesian network, we evaluate two initialization strategies: one where the network is initialized with deterministic weights, as described in section 3, and another is randomly initialized, with both the μ and ρ trainable parameters set to random values. By training a ResNet18 on CIFAR10 on all three models, we find that our method accelerates convergence over randomly initialized 5% network, with an equal number of variational parameters and distribution over the network, and over standard full VI network, achieving a lower NLL and higher accuracy at a significantly lower training cost (see Figure 7).

We further evaluate the scalability of our method on the ImageNet classification task Deng et al. (2009) using ResNet50 He et al. (2016). In contrast to full VI Ovadia et al. (2019) and linearized Laplace approximations



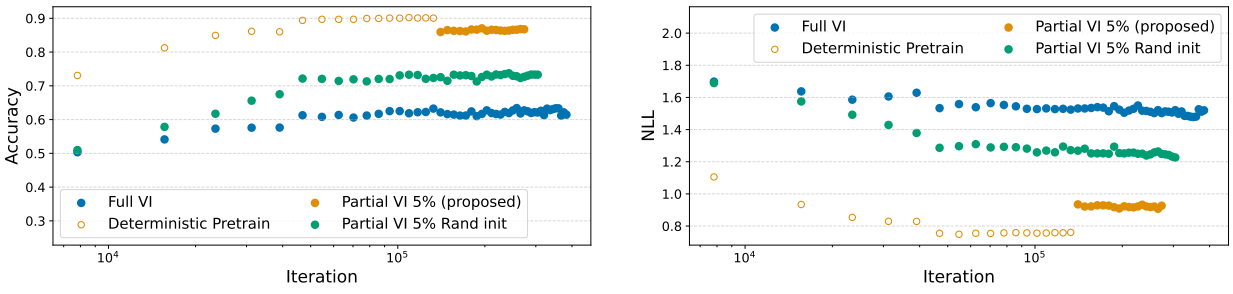
(a) ΔError ↓ (b) ΔNLL ↓ (c) Correctly Classified (d) Incorrectly Classified

Figure 6: Results for ImageNet with respect to covariate shift performance. (a) error, (b) NLL relative to uncorrupted performance, with Partial 1% displaying more robustness to covariate shifts. (c) Entropy values for correctly vs. incorrectly classified inputs for uncorrupted and corrupted data for Partial 1% model, displaying calibration under covariate shift, with increasing entropy (uncertainty) with increasing data corruption. Ensemble and Dropout results shown are pulled from Ovadia et al. (2019)

Daxberger et al. (2021), which face challenges in scaling to large datasets, our proposed partial Bayesian approach with low Bayesian rates offers an effective trade-off between computational efficiency and reliable uncertainty estimation. As shown in Figure 6, our method maintains robustness under covariate shift on large-scale data, outperforming ensembles and dropout (Figures 6a and 6b). Additionally, Figure 6d demonstrates that our model exhibits increased predictive entropy with higher levels of corruption on ImageNet-C Hendrycks & Dietterich (2019), particularly for misclassified inputs—indicating well-calibrated uncertainty under distributional shift.

4.4 Segmentation on CityScapes

We demonstrate our method on a large-scale, complex task, specifically multi-class pixel-level segmentation. Due to the high computational demands, general approaches for estimating uncertainty in segmentation are often limited to approximations such as ensembles and Monte Carlo dropout Abdar et al. (2021) rather than variational Bayesian methods. Our first-order gradient selection criterion and PBN training enable seamless segmentation and uncertainty estimation. This contrasts with the method proposed by Daxberger et al.



(a) Accuracy (b) NLL

Figure 7: Test accuracy (a) and NLL (b) as a function of number of training iterations, for a full variational inference network (VI), a partial Bayesian network with 5% Bayesian parameters initialized from deterministic weights (Partial VI 5% (proposed)), and another 5% partial Bayesian network initialized from random (Partial VI 5% Rand init). (ResNet18 on CIFAR10)

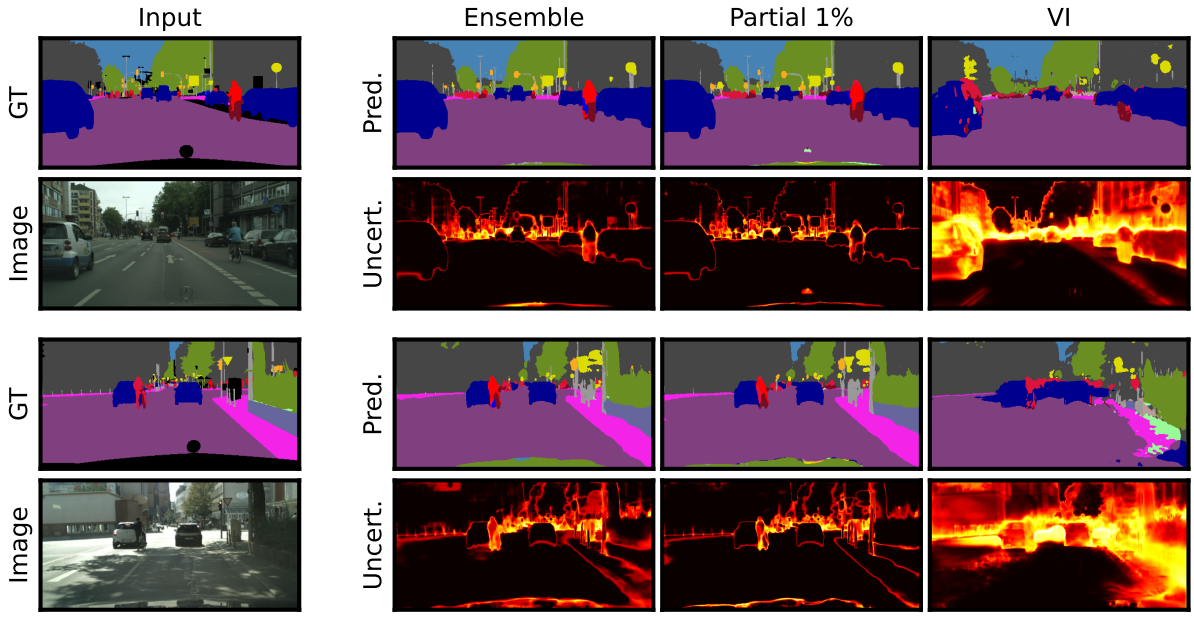


Figure 8: Cityscapes pixel-wise segmentation samples: from left to right: ground truth segmentation, segmentation and uncertainty for ensemble, Partial 1% (ours), and full VI model. Low uncertainty is depicted as dark, while yellow-red indicates higher uncertainty. Note: the VI model is trained for an additional 4-fold number of iterations. Uncertainty is measured by entropy. (see additional qualitative examples in Appendix:10)

(2021), which utilizes linearized Laplace and selects the subnetwork based on the second-order Hessian. This process is computationally prohibitive for complex segmentation applications. To demonstrate our method on Cityscapes segmentation, we use a vanilla UNet architecture Ronneberger et al. (2015), with 5 downward and upward convolutional blocks, for a total number of parameters of over 31.38M (to put in perspective, ResNet50 has 25.56M parameters). We compare the performance of the deterministic, VI, 5-member ensemble, and PBN with $r_{bayes} = 1\%$. Figure 8 shows the qualitative comparison of the ensembles, PBN-1%, and VI segmentation performance. Qualitatively, the PBN-1% segmentation uncertainty consistent with errors in the prediction and at class boundaries. PBN-1% is also consistent with uncertainty estimations by ensembles, while utilizing 80% fewer trainable parameters. Quantitatively, PBN has less than 5% degradation in IoU performance compared to the deterministic model, while the full VI model performs very poorly with a 40% degradation in performance 9. The performance of the PBN on the segmentation task is both quantitatively and qualitatively comparable to that of ensembles. Examining the distribution of selected parameters in the subnetwork, Top$k|\nabla_\theta|$ at low $r_{bayes} < 5\%$ aligns with Sharma & Jennings (2021), showing that the highest magnitude gradients are near the network edges, both input and output. For ResNet18 with CIFAR-10, Bayesian weights are selected from layers near the input and output blocks, consistent with Sharma & Jennings (2021) findings. The distribution of the variational parameters as a function of layer depth is shown in Appendices 12 and 13.

5 Scope and Limitations

Our method relies on pre-trained point estimates to initialize the partial Bayesian networks, similar to Daxberger et al. (2021), however applied in the context of variational inference as opposed to Linearized Laplace. While the partial Bayesian formulation effectively reduces parameter complexity in large probabilistic models, its advantages diminish for smaller networks. As demonstrated in the toy example in Section 4.1, the added computational overhead may outweigh the benefits, particularly for uncertainty estimation in

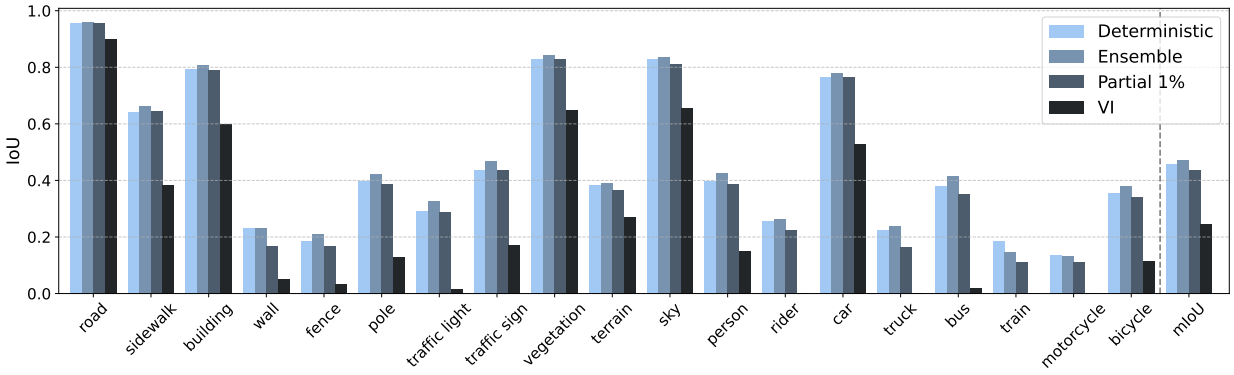


Figure 9: Comparative results on Cityscapes: per class IoU and mean IoU for each method. Partial method proposed with 1% variational parameters performing within 4% of Deterministic performance and 7% of Ensembles. On the other hand variational Bayesian method with 4-fold additional FLOPs performs 43% worse than Partial 1% model and 47% worse than Ensembles.

out-of-distribution (OOD) regions. For small to medium-sized models, ensembles remain more effective for both in-distribution (ID) and OOD settings, as their relative computational cost is negligible.

A current limitation of our approach is the reliance on posterior sampling via multiple forward passes (N-samples) to estimate uncertainty, as the method does not yet address the posterior sampling challenge inherent to Bayesian neural networks. Recent work on sampling-free variational Bayesian last layers Harrison et al. (2024) introduces a more deterministic alternative for this issue. In parallel, we are actively exploring improvements to reduce the sampling-related training and inference FLOPs in our implementation.

Currently, our method supports linear and convolutional layers, making it suitable for regression, classification, and segmentation tasks. Extension to recurrent architectures remains an open direction for future work.

6 Conclusion

We introduced a scalable algorithm for training partial Bayesian neural (PBN) networks using variational inference. Starting from a pre-trained deterministic model, our method selects a subnetwork of parameters to treat as Bayesian, guided by first-order gradients obtained through standard automatic differentiation—without additional computational overhead. This approach enables control over the degree of Bayesian modeling via a simple hyperparameter.

For small- to medium-scale networks (10^4–10^5 parameters), ensembles remain more effective in delivering well-calibrated predictive uncertainty. However, in larger models (10^6–10^7 parameters), our approach provides a cost-effective alternative for uncertainty quantification. With fewer than 5% of parameters designated as Bayesian, our approach achieves up to a 50% reduction in trainable parameters compared to a full VI network and up to 80% compared to ensembles—while maintaining reliable uncertainty estimates.

Partial (sparse) VI demonstrates strong robustness to data corruption and outperforms or matches established techniques such as ensembles, dropout, and linearized Laplace approximations Daxberger et al. (2021). Additionally, we validate the scalability of our approach on large-scale tasks, including classification on ImageNet and semantic segmentation, where traditional Bayesian neural network methods (e.g., VI or MCMC) often become infeasible. This work opens new avenues for efficient and scalable uncertainty estimation in deep learning, particularly in safety-critical domains where full Bayesian modeling remains impractical.

References

Zeinab Abboud, Herve Lombaert, and Samuel Kadoury. Sparse bayesian networks: Efficient uncertainty quantification in medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 675–684. Springer, 2024.

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1): 1–3, 1950.

Shiye Cao, Anqi Liu, and Chien-Ming Huang. Designing for appropriate reliance: The roles of ai uncertainty presentation, initial user decision, and user demographics in ai-assisted decision-making. *Proceedings of the ACM on Human-Computer Interaction*, 8(CSCW1):1–32, 2024.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, pp. 2510–2521. PMLR, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Sourya Dey, Kuan-Wen Huang, Peter A. Beerel, and Keith M. Chugg. Pre-defined sparse neural networks with hardware acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9 (2):332–345, 2019.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Martin Ferianc, Partha Maji, Matthew Mattina, and Miguel Rodrigues. On the effects of quantisation on model uncertainty in bayesian neural networks. volume 161, pp. 929–938. PMLR, 27–30 Jul 2021.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

James Harrison, John Willes, and Jasper Snoek. Variational bayesian last layers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pp. 1861–1869. PMLR, 2015.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pp. 1169–1179. PMLR, 2020.

Xiaotao Jia, Jianlei Yang, Runze Liu, Xueyan Wang, Sorin Dan Cotofana, and Weisheng Zhao. Efficient computation reduction in bayesian neural networks through feature decomposition and memorization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1703–1712, 2021.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, volume 28, 2014.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Harry H Ku et al. Notes on the use of propagation of error formulas. *Journal of Research of the National Bureau of Standards*, 70(4), 1966.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

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

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *NeurIPS*, volume 2, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 1998.

David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32, 2019.

Andrey Malinin, Neil Band, German Chesnokov, Yarin Gal, Mark JF Gales, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, et al. Shifts: A dataset of real distributional shift across multiple large-scale tasks. *arXiv preprint arXiv:2107.07455*, 2021.

Andrey Malinin, Andreas Athanasopoulos, Muhamed Barakovic, Meritxell Bach Cuadra, Mark JF Gales, Cristina Granziera, Mara Graziani, Nikolay Kartashev, Konstantinos Kyriakopoulos, Po-Jui Lu, et al. Shifts 2.0: Extending the dataset of real distributional shifts. *arXiv preprint arXiv:2206.15407*, 2022.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *NeurIPS*, volume 1, 1988.

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

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

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Snehal Prabhudesai, Jeremiah Hauth, Dingkun Guo, Arvind Rao, Nikola Banovic, and Xun Huan. Lowering the computational barrier: Partially bayesian neural networks for transparency in medical imaging ai. *Frontiers in Computer Science*, 5, 2023.

Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. Ai in health and medicine. *Nature medicine*, 28(1):31–38, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015.

Himanshu Sharma and Elise Jennings. Bayesian neural networks at scale: a performance analysis and pruning study. *The Journal of Supercomputing*, 77:3811–3839, 2021.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic? In *International Conference on Artificial Intelligence and Statistics*, pp. 7694–7722. PMLR, 2023.

Jun Shi, Jianfeng Xu, Kazuyuki Tasaka, and Zhibo Chen. Sasl: Saliency-adaptive sparsity learning for neural network acceleration. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(5): 2008–2019, 2020.

Mahesh Subedar, Ranganath Krishnan, Sidharth N Kashyap, and Omesh Tickoo. Quantization of bayesian neural networks and its effect on quality of uncertainty. In *Workshop on Uncertainty and Robustness in Deep Learning, International Conference on Machine Learning*, 2021.

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

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, 2017.

Kai Yang, Xiaolin Tang, Jun Li, Hong Wang, Guichuan Zhong, Jiaxin Chen, and Dongpu Cao. Uncertainties in onboard algorithms for autonomous vehicles: Challenges, mitigation, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

Daniel S Yeung, Ian Cloete, Daming Shi, and Wing wY Ng. *Sensitivity analysis for neural networks*. Springer, 2010.

Jiaming Zeng, Adam Lesnikowski, and Jose M Alvarez. The relevance of bayesian layer positioning to model uncertainty in deep bayesian active learning. In *Third workshop on Bayesian Deep Learning in Advances in Neural Information Processing Systems*, 2018.

J.M. Zurada, A. Malinowski, and I. Cloete. Sensitivity analysis for minimization of input data dimension for feedforward neural network. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, volume 6, pp. 447–450, 1994.

A Additional Segmentation Results

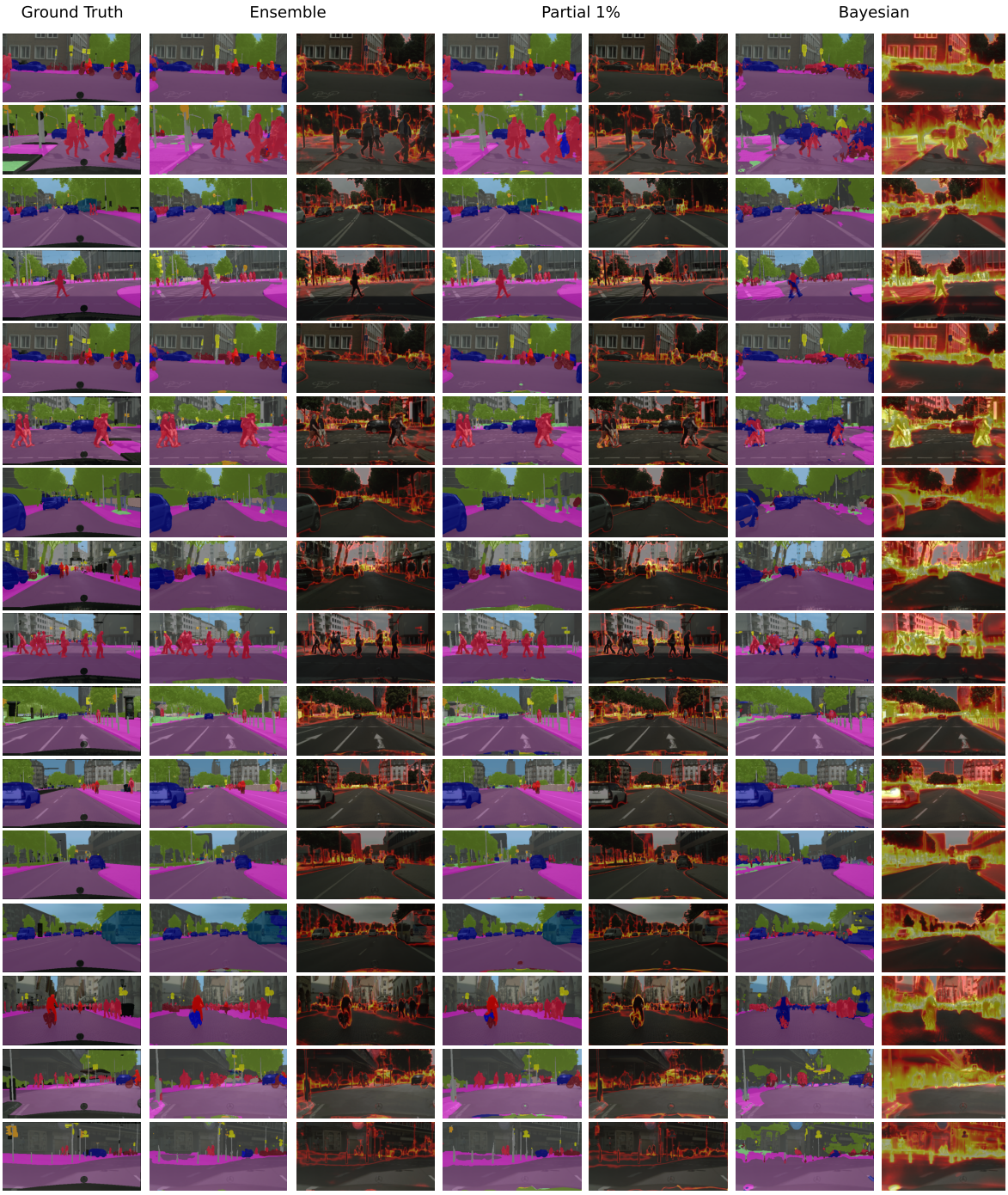


Figure 10: Cityscapes fine-segmentation samples: from left to right: ground truth segmentation, segmentation and uncertainty for ensemble, 1% PBN (ours), and full VI model (Bayesian). Low uncertainty is depicted as dark overlays, while yellow-red indicates higher uncertainty. Note: the full VI network is trained for an additional 4-fold Epochs.

Table 1: Compute and memory costs relative to the deterministic model. N denotes the number of posterior samples, f_d denotes the FLOPs required for a single forward pass, M is the number of ensemble members, and m is the memory required to store a deterministic model ($m \sim f_d$) Ovadia et al. (2019)

Model	**#Parameters**	**Relative FLOPS**	**Memory**
Deterministic	θ	$3f_d$	m
M-Ensemble	$M\theta$	$3Mf_d$	M×m
VI	2θ	$2 \times (N+2)f_d$	2×m
PBN	$(1+r_{bayes})\theta$	$3f_d \times \text{epochs}_{pretrain} + (1+r_{bayes})(N+2)f_d$	2×m

B Parameter Complexity

Given a deterministic model of θ parameters as a baseline. A fully Bayesian model increases the number of parameters to 2θ, where each layer is characterized by two distinct parameters, accounting for the distribution's mean and standard deviation. For models employing partial Bayesian techniques, a singular Bayesian layer results in a parameter count of $\theta + L_{numel} < 2\theta$ where L_{numel} represents the number of elements in a parameter layer. In partial Bayesian models with n Bayesian weights, the parameter count becomes $\theta + n = (1 + r_{bayes})\theta$.

C FLOPs Count

The floating point operations (FLOPs) are computed as the sum of tensor additions and multiplications. The FLOPs for a forward pass include calculating the loss for a single batch for the given set of parameters θ_i at epoch i. In the backward pass, the loss is used to calculate the gradients of the parameters $\nabla\theta_i$ and the gradient of the activations. Therefore, to account for the total FLOPs for a single forward-backward pass, it would cost $3f_d$ for a single sample, where f_d is the number of FLOPs for a fully-dense deterministic forward pass for a given architecture. For a given architecture, we can compute the following:

- **Ensemble** Given $M-$ensemble members, the cost for a single sample scales with $3 \times M \times f_d$.
- **VI** Given $N-$posterior samples, the cost for a single sample scales with $2 \times N \times f_d + 2(2 \times f_d)) = 2 \times f_d(N+2)$, where the $2\times$ term accounts for the addition of the uncertainty σ parameter.
- **Partial Bayesian** Given a Bayesian rate $0 < r_{bayes} < 1$ the number of FLOPs for the forward method is $(1+r_{bayes}) \times N \times f_d$, where N is the number of posterior samples, and backward method costs $2\times(1+r_{bayes})\times f_d$ for a total of $(1+r_{bayes})f_d(N+2)+3f_d\times\text{epochs}_{\text{pretrain}}$ where the second term accounts for the deterministic pre-training step. FLOPs associated with the intermediate sensitivity analysis step (algorithm 1 line 8) are not taken into account as the contribution to the total FLOPs is negligible.

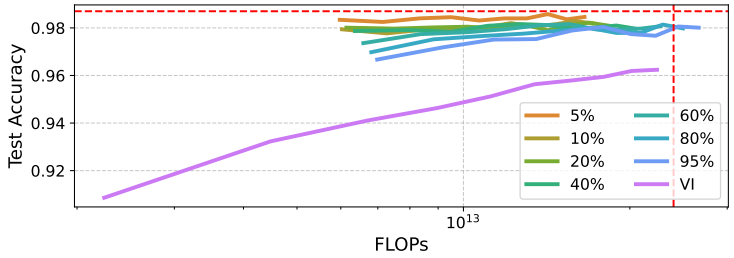
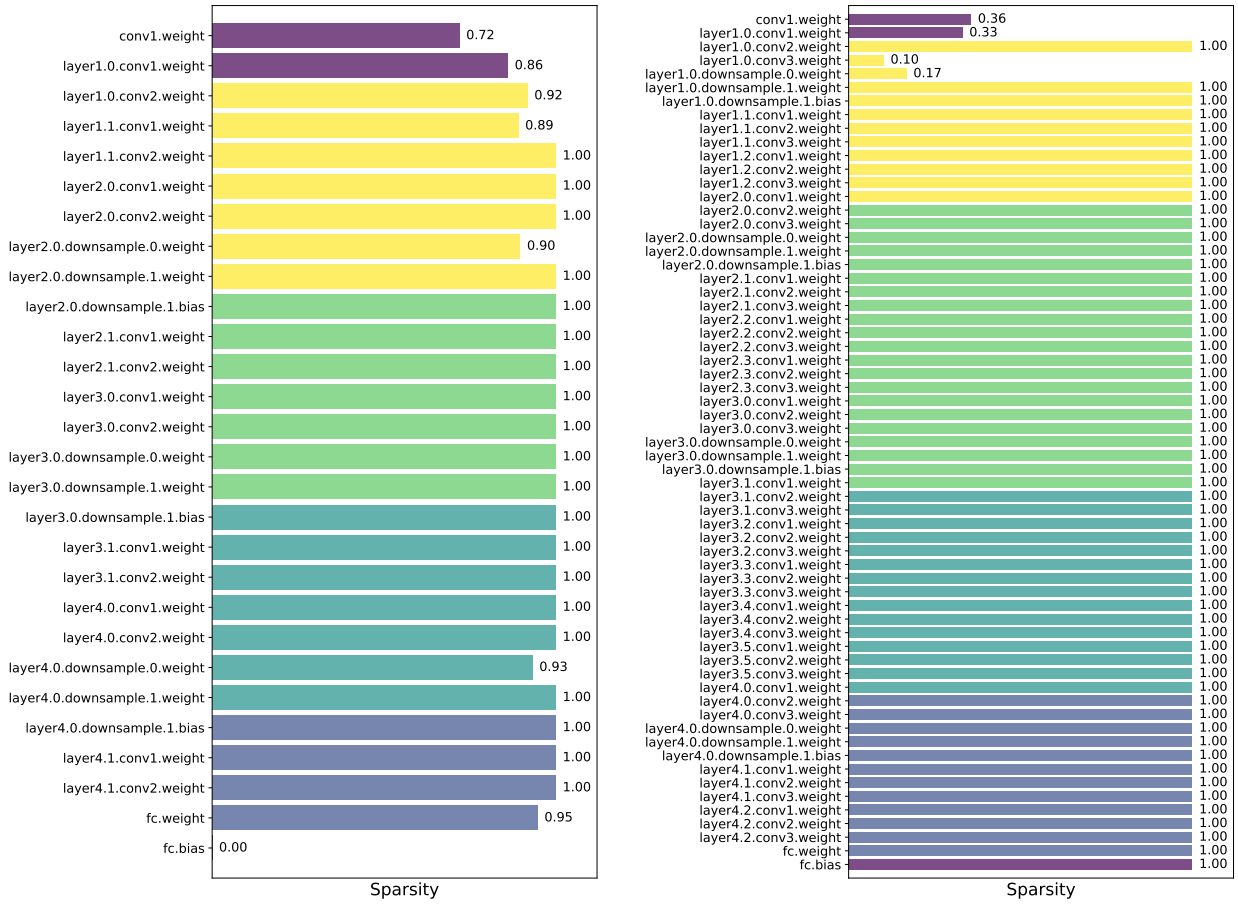


Figure 11: Test Accuracy vs. training FLOPs with the ensemble performance highlighted with the intersection of the dotted red lines for MNIST. The Legend = r_{bayes} values, and Bayes for the fully Bayesian model.

D Sparse Bayesian Parameter Distribution - ResNet

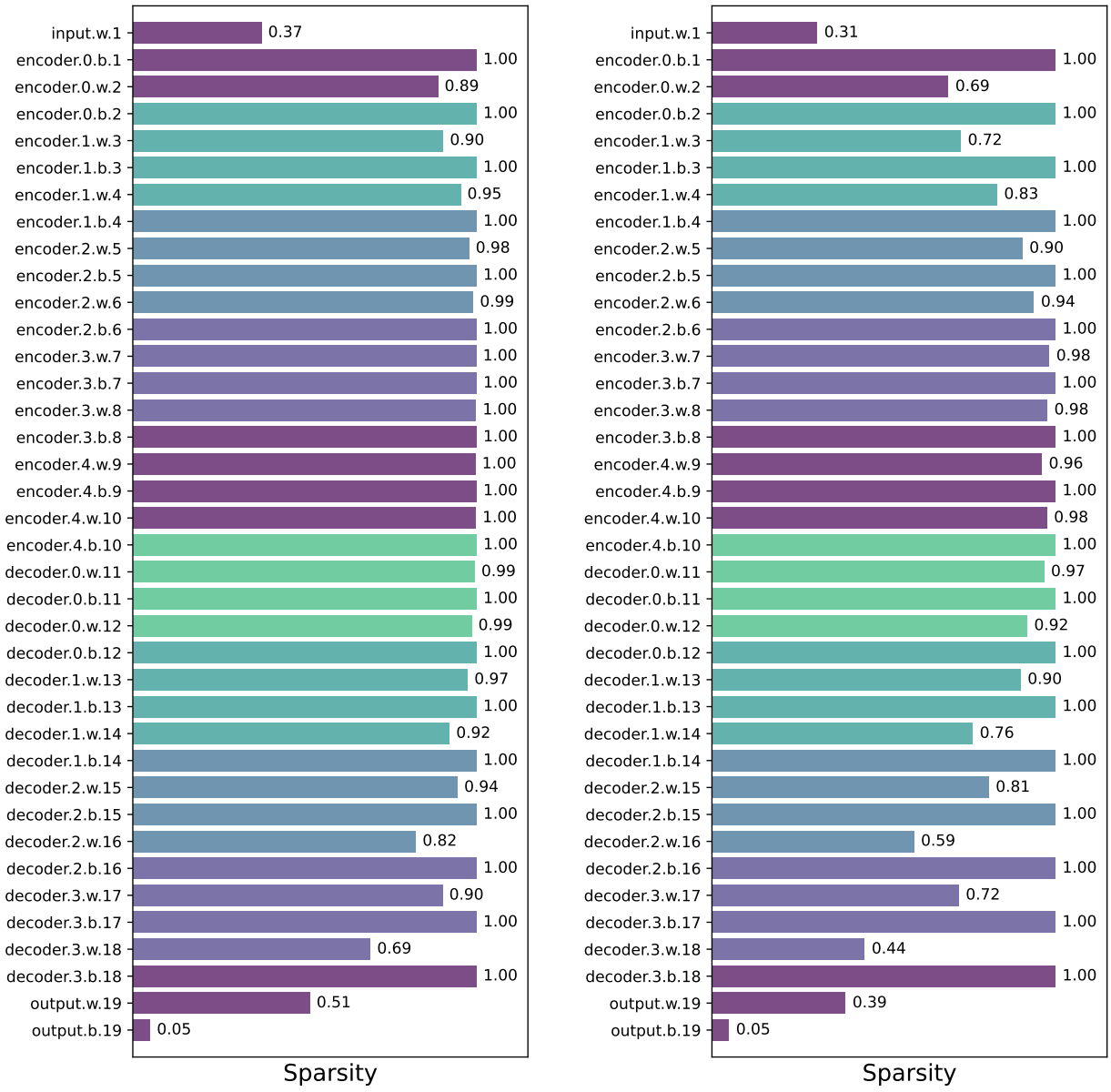


(a) CIFAR10-ResNet18-1%

(b) ImageNet-ResNet50-1%

Figure 12: Sparsity values of ResNet18(50) layers with $r_{bayes} = 1\%$]. Layers with a sparsity value of 0.99-1.0 are fully deterministic.

E Sparse Bayesian Parameter Distribution - UNet

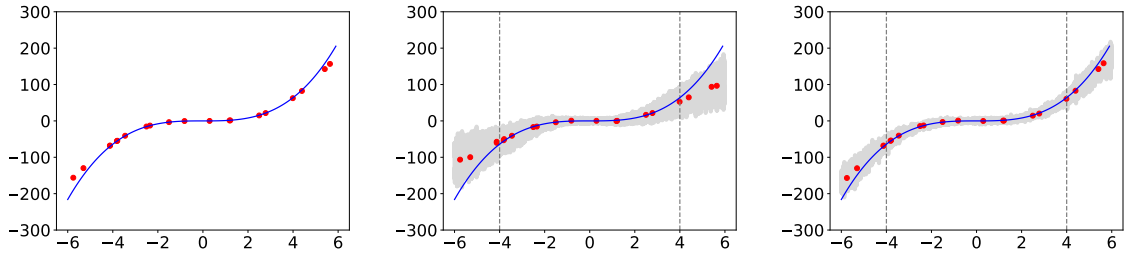


(a) UNet-1%

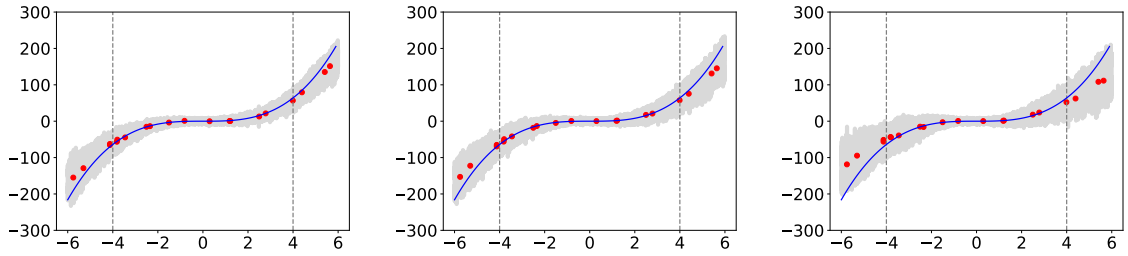
(b) UNet-5%

Figure 13: Sparsity values of UNet with encoder/decoder blocks of feature [64, 128, 256, 512, 1024] layers with $r_{bayes} = [1\%, 5\%]$. Layers with a sparsity value of 0.99-1.0 are fully deterministic.

F Toy Regression Dataset



(a) Deterministic (b) VI (c) $r_{bayes} = 10\%$

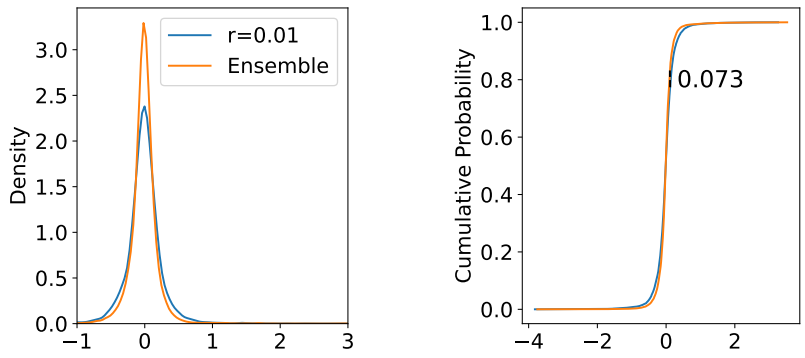


(d) $r_{bayes} = 20\%$ (e) $r_{bayes} = 40\%$ (f) $r_{bayes} = 80\%$

Figure 14: Results on $y = x^3 + \epsilon$ toy dataset. (a-d) Predictions on toy data, the blue line is ground truth $y = x^3$, red dots are sample test predictions, and the gray shaded area is the 3σ confidence interval. Predictions on (a) Deterministic, (b) Full VI Bayesian approach, (c-f) Partial Bayesian models with different Bayes rates. Note that the dotted lines highlight the in- and out-of-distribution test points. The partial Bayes model displays how varying r_{bayes} allows the model to capture the predictive uncertainty for in- and -out of distribution samples.

G Learned Weight Distribution

We evaluated the weight distribution using a Gaussian Mixture Model (GMM) for the 5-member ensemble and PBN-1%. Figure 15a shows that the ensemble's weight distribution is slightly narrower than that of the PBN-1%. A Kolmogorov–Smirnov (KS) test was used to quantify the difference between the two GMM distributions. Figure 15b plots the cumulative distribution functions (CDFs) of the weight GMMs. The KS statistic for PBN-1% compared to the 5-ensemble is 0.07 (with p-value $< 1e-10$), indicating a high similarity between the two distributions.



(a) (b)

Figure 15: (a) Gaussian Mixture Model (GMM) for CIFAR10 ResNet models from PBN-1%, 5-member ensemble. (b) Kolmogorov–Smirnov (KS) cumulative distribution function for PBN-1% with a KS statistic of 0.07 and a p-value $< 1e-5$. The KS-test provides supporting evidence for the similarity of the learned models by assessing the similarity of their weight distributions.

H Additional Model and Training Details

H.1 Hardware and Software

Models were trained with an NVIDIA RTX A6000 GPU with Pytorch 2.0.1 version.

H.2 Model Details

H.2.1 MNIST

Standard LeNet 300-100 network was used for the MNIST classification experiments with a batch size of 50. SGD optimizer with a fixed learning rate of 0.01, a momentum of 0.9, and an L2 regularization coefficient of 10^{-5}. ReLU non-linearities were used for all models and batch normalization. Models trained for 50 epochs.

Table 2: List of datasets used in the paper, with the train/test splits.

Dataset	Source	Size (Train, Test)	Input Size	#Classes
MNIST	LeCun (1998)	60K , 10K	28×28	10
Fashion-MNIST	Xiao et al. (2017)	60K, 10K	28×28	10
CIFAR10	Krizhevsky et al. (2009)	50K, 10K	32×32	10
CIFAR10-C	Hendrycks & Dietterich (2019)	10K	32×32	10
SVHN	Netzer et al. (2011)	732K, 260K	32×32	10
CityScapes	Cordts et al. (2016)	4500, 500	256×512	20

H.3 CIFAR-10/SVHN

H.3.1 Small ResNet

A small ResNet with 4 ResBlock was used for initial experiments, containing features of [16, 32, 64]. Each ResBlock contains 2 convolutional layers, one with kernel size 3, the first with a stride of 2 and the second convolutional layer with a stride of 1. The input convolution is of kernel size 3 and a stride of 2, followed by a Maxpool layer of (2,2). Similarly, as above, a batch size of 50, an SGD optimizer with a fixed learning rate of 0.01, momentum of 0.9, and L2 regularization coefficient of $10^{-}5$. Models were trained for 50 epochs. Both CIFAR10 and SVHN datasets were trained with $r_{bayes} = [0.1\%, 0.5\%, 1\%, 5\%, 10\%, 20\%, 40\%, 60\%, 80\%, 95\%$.

H.3.2 Standard ResNet18-V1

Our second CIFAR10 model used a standard ResNet18-V1 He et al. (2016) with ReLU activations. The max pool layer was removed, and the initial convolution of 7x7 of stride 2 was changed to 7x7 with a stride of 1 to avoid significant downsampling of the smaller CIFAR10 images. The model was trained with a batch size of 200 for 50 epochs using an SGD optimizer with a momentum of 0.9 and a learning rate schedule that includes a linear warm-up for up to 5 epochs to a maximum learning rate of 1.6. Then, the learning rate is reduced by 10 at epoch 30. L2 regularization coefficient of 10^{-4} was used. For the Bayesian and Partial Bayesian approaches, the models were trained for 100 epochs. A dropout rate of 0.2 was used to avoid overfitting; dropout layers were added after the FC layer and after the first convolutional layer in each convblock.

H.4 CityScapes

We use a standard UNet Ronneberger et al. (2015) with standard encoder-decoder architecture, convolutional blocks with feature output sizes of [64, 128, 256, 512, 1024]. The upward paths between each block in the decoders are bilinear interpolations due to a limitation in the Pytorch backward pass customization for TransposeConvolutional layers. The deterministic models were trained with SGD optimizer, a momentum of 0.9, and an initial learning rate of 0.1, which is reduced by 0.5 on plateauing validation performance. The models were trained for 150 epochs. The data was resized to 256×512. A weighted cross-entropy loss was used to train the dataset to balance the minority classes, such as persons, riders, trailers, motorcycles, caravans, trains, etc. We follow the recommended data labeling setup by the dataset initial release Cordts et al. (2016).

H.5 Ensembles

Five member ensembles were used in our experiments, where each member was trained independently with a different random seed [0,1,2,3,4] to ensure reproducibility.

H.6 Variationa Inference

For all Partial Bayesian training or full VI training -ELBO loss is minimized, where $\mathcal{L} = \mathcal{L}(f_{\Theta_d}(x_i), y_i) + \beta \cdot KL(\mathcal{N}_{prior}, \mathcal{N}_{post})$, the first term is the MLE term (for regression it's MSE for classification it's cross-entropy loss), β is the weight of the KL divergence term. The KL weight term, β, is annealed during training epochs, transitioning smoothly from an initial value [0.2] to a desired target value [0.01] as a function of epochs $\beta_i = \beta_{target} - (\beta_{target} - \beta_{init}) \times (epoch_i / epoch_{total})$. We experimented with fixed vs. different annealing schedules and found this schedule optimal to minimize the KL divergence in the first few epochs and then optimize the task-specific loss. 5 posterior samples were drawn during training and inference.