DUALITY OF INFORMATION FLOW: INSIGHTS IN GRAPHICAL MODELS AND NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

This research highlights the convergence of probabilistic graphical models and neural networks, shedding light on their inherent similarities and interactions. By interpreting Bayesian neural networks within the framework of Markov random fields, we uncovered deep connections between message passing and neural network propagation. Our exploration unveiled a striking equivalence between gradients in neural networks and posterior-prior differences in graphical models. Empirical evaluations across diverse scenarios and datasets showcased the efficacy and generalizability of our approach. This work introduces a novel perspective on Bayesian Neural Networks and probabilistic graphical models, offering insights that could pave the way for enhanced models and a deeper understanding of their relationship.

1 INTRODUCTION

Probabilistic graphical models and neural networks are two distinct paradigms for modeling data generation within networks composed of fundamental computational units. A probabilistic graphical model defines the joint probability distribution of a network of random variables by leveraging conditional probabilities or clique potentials. In contrast, a neural network characterizes the transformation of a tensor "particle" as it progresses through multiple layers, encompassing both linear and nonlinear operations, to achieve specific outcomes defined by a loss function. In a broader context, each computational graph, trained with a dataset sampled from a data distribution via stochastic gradient descent, gives rise to a probabilistic graphical model. This model delineates a joint probability distribution involving synaptic weights and the data. Conversely, every probabilistic graphical model leads to a computational graph in which stochastic message passing strives to attain detailed balance, ultimately resulting in a stationary joint probability distribution. While simulation-based approaches are often more scalable for processing extensive datasets when compared to analytical methods, the exploration of connections between deterministic and stochastic perspectives within network-based computational models holds the promise of yielding enhanced models and deeper insights (Bishop, 2006; Goodfellow et al., 2016).

In our research, we conceptualized a Bayesian neural network as a Markov random field (Neal, 1995), where the loss function serves as potential energy governing the trajectories of tensors as they navigate through the computational graph and interact with synaptic weights. The focus was on identifying mean parameters of tensors during forward propagation and computing gradients with respect to their canonical parameters in backward propagation (Rockafellar, 1970; Wainwright & Jordan, 2008; Khan & Rue, 2021). This process reveals the gradient as the difference in mean parameters between the posterior and prior distributions, highlighting a parallel between probabilistic graphical models' message-passing and neural networks' forward/backward propagation. Furthermore, we formulated the mean parameters during forward propagation and their sensitivities during backward propagation in terms of the statistics of tensor particles as they advanced through the computational graph and the gradients of loss with respect to these tensors as they propagated backward within the graph. This approach sheds light on the duality between probabilistic graphical models and Bayesian neural networks, linking population dynamics and distributions. In our paper, we use the term "particle" to describe a set comprising inputs, post-activations, synaptic weights, and labels that collectively constitute the state of a Bayesian neural network. This terminology is borrowed from Monte Carlo methods prevalent in statistical physics

and computational mathematics, where a particle represents a possible state of the system under study, analogous to how physical particles behave and interact (Gardiner et al., 1985).

Our algorithm underwent extensive testing across various datasets including CIFAR 10/100, Tiny ImageNet, and UCI regression, using neural network architectures like DenseNet, ResNet, and others. We explored different learning algorithms, such as Bayes-by-backprop for Bayesian neural networks (BNNs), and incorporated techniques like cosine learning rate scheduling and image augmentation. We also addressed vanishing gradient issues in BNNs using the variational message passing algorithm (Winn et al., 2005) and showcased BNNs' superior generalization through test error analysis and visualization of learned data distributions.

2 NOTATION

A neural network, as explained in Goodfellow et al. (2016), emulates decision-making processes based on a training dataset D by minimizing a specific loss. This loss is essentially the empirical mean of the loss function $J(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}), \mathbf{y})$, calculated in the following manner:

$$\operatorname{argmin}_{W=\{\mathbf{W}_1 \ldots, \mathbf{W}_L\}} \frac{1}{|D|} \sum\nolimits_{(\mathbf{x},\mathbf{y})\in D} J\left(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}), \mathbf{y}\right), \tag{1}$$

$$\text{where } \hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}) = \mathbf{x}_L, \mathbf{x}_l = \mathbf{f}_l(\mathbf{a}_l), \mathbf{a}_l = \mathbf{W}_l \cdot \mathbf{x}_{l-1} + \mathbf{b}_l \text{ for } l = 1, \ldots, L, \text{ and } \mathbf{x}_0 = \mathbf{x}.$$

In these equations, $\mathbf{a}_l$, $\mathbf{x}_l$, $\mathbf{W}_l$, and $\mathbf{f}_l$ denote the pre-activation, post-activation, weights, and activation function of each layer l, respectively. The size of the dataset $|D|$ is the count of training examples. Often, the loss function J is the negative log-likelihood, $J(\hat{\mathbf{y}}, \mathbf{y}) = -\log p(\mathbf{y}; \hat{\mathbf{y}})$, capturing the cross-entropy between the empirical distribution of training data and the probabilistic model. In our notation, operators are applied from the left, meaning $\mathbf{W}_l \cdot \mathbf{x}_{l-1} = \left(\sum_{j_{l-1}} \mathrm{W}_{i_l, j_{l-1}} \mathrm{x}_{j_{l-1}}\right)_{i_l}$ computes pre-activation elements at multi-index i_l as the weighted sum of post-activation elements at multi-index j_{l-1}. For ease of understanding, multi-indices can be treated as standard integer indices, with synaptic weights represented as matrices and post-activation tensors as vectors. The dimensionality of these indices depends on the layer's architecture.

Let $\boldsymbol{\delta}_l = \nabla_{\mathbf{a}_l} J(\hat{\mathbf{y}}, \mathbf{y})$ denote the gradient of the loss with respect to the activation inputs at layer l, commonly referred to as the sensitivity of activation inputs (Stork et al., 2000). Backpropagation recursively computes these gradients for activation inputs and weights through automatic differentiation: $\boldsymbol{\delta}_L = \mathbf{f}'_L \circ \nabla_{\hat{\mathbf{y}}} J(\hat{\mathbf{y}}, \mathbf{y})$, $\boldsymbol{\delta}_{l-1} = \mathbf{f}'_{l-1} \circ \left(\mathbf{W}_l^\top \boldsymbol{\delta}_l\right)$, and $\nabla_{\mathbf{W}_l} J(\hat{\mathbf{y}}, \mathbf{y}) = \boldsymbol{\delta}_l \mathbf{x}_{l-1}^\top$, for l ranging from L to 1. Here, ∇ represents the gradient, $\circ$ indicates element-wise multiplication, $\bullet^\top$ signifies matrix transpose, and $\mathbf{f}'_l$ represents the derivative of the activation function.

A Bayesian neural network (Neal, 1995) mimics decision-making behavior in training data using an ensemble of neural networks that share the same computational graph but have distinct synaptic weights. In this paper, we frame the learning problem as the minimization of the following variational principle over a set of variational posterior parameters $\boldsymbol{\theta}$:

$$\mathbf{E}_{q(\mathbf{W};\boldsymbol{\theta}_\mathbf{W})} \sum\nolimits_{(\mathbf{x},\mathbf{y})\in D} J\left(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}), \mathbf{y}\right) + \log q(\mathbf{W}; \boldsymbol{\theta}_\mathbf{W}) / p(\mathbf{W}). \tag{2}$$

Here, $\mathbf{W} = \{\mathbf{W}_1 \ldots, \mathbf{W}_L\}$ represents the weights, $p(\mathbf{W})$ is the prior probability, and $q(\mathbf{W}; \boldsymbol{\theta}_\mathbf{W})$ is the variational posterior probability. All other symbols are consistent with those of a non-Bayesian neural network. The objective is to minimize the negative evidence lower bound. The stochastic weights introduce a probability distribution over $\hat{\mathbf{y}}(\mathbf{x})$. The ensemble decision is formulated as Bayesian model averaging $\mathbf{E}_{q(\mathbf{W};\boldsymbol{\theta}_\mathbf{W})} \hat{\mathbf{y}}(\mathbf{x}|\mathbf{W})$, and model uncertainty can be assessed through the entropy of the ensemble decision $-\mathbf{E}_{q(\mathbf{W};\boldsymbol{\theta}_\mathbf{W})} \log p(\hat{\mathbf{y}}|\mathbf{x}, \mathbf{W})$.

Minimizing the variational free energy in the aggregated loss $-\sum_{(\mathbf{x},\mathbf{y})\in D} \log p(\mathbf{y}|\mathbf{x})$ concurrently minimizes the variational free energy in the negative log-likelihood of the training data $-\sum_{(\mathbf{x},\mathbf{y})\in D} \log p(\mathbf{x}, \mathbf{y})$, because $\log p(\mathbf{x})$ is independent of $\mathbf{W}$. Therefore, optimizing Eq. 2 with respect to the variational parameters $\boldsymbol{\theta}_\mathbf{W}$ provides a sampling-free methodology (Tieleman & Hinton, 2009; Friston, 2010; Lee & LeCun, 2017; Song et al., 2021) for learning complex probability distributions over the data D. Here, the potential energy is defined as $\log p(\mathbf{x}, \mathbf{y}, \mathbf{W}) = J(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}), \mathbf{y}) - \log p(\mathbf{W}) + \text{constant}$.

As tensor "particles" traverse multiple layers of linear and nonlinear transformations within an ensemble of neural networks during forward propagation, their probability distribution evolves accordingly. Likewise, during backpropagation, as the loss gradient with respect to these tensors traverses these transformation layers, it characterizes the sensitivity of the probability distributions. This sensitivity, in turn, generates various probability kernels that guide the particles and modify the probability distributions toward minimizing the loss. In this context, a duality (Gardiner et al., 1985) emerges between the stochastic tensor flow within a Bayesian neural network and the deterministic evolution of probability distributions within a probabilistic graphical model.

The case of particular interest is when the data-generating process from $\mathbf{x}_0$ to $\mathbf{x}_{l=1,\dots,L}$ and $\mathbf{y}$ is approximately Gaussian-linear. As suggested by Jacot et al. (2018), neural networks tend to remain approximately linear throughout training in overparameterized regimes. This linear approximation simplifies the analysis of convergence and generalization, as it essentially involves inferences using multivariate normal distributions. In such scenarios, the activation function can be approximated using its first-order Taylor expansion. Additionally, the synaptic weights are reparameterized using weight parameters $\boldsymbol{\theta}_{\mathbf{W}_l}$ and a standard Gaussian random vector $\boldsymbol{\omega}_l$. This leads to a Gaussian Markov random field, as described by Rue & Held (2005), characterized by the following Langevin process (describing the stochastic tensor flow) and the Fokker-Planck process (detailing the tensor distribution evolution):

$$
\begin{aligned}
&\mathbf{x}_l = \mathbf{f}_l(\mathbf{x}_{l-1}, \mathbf{W}_l) \approx \hat{\mathbf{x}}_l + \hat{\mathbf{f}}'_{\mathbf{x}_{l-1}} \cdot (\mathbf{x}_{l-1} - \hat{\mathbf{x}}_{l-1}) + \hat{\mathbf{f}}'_{\mathbf{W}_l} \cdot \left(\mathbf{W}_l - \hat{\mathbf{W}}_l\right), \\
&\mathbf{y} = \mathbf{g}(\mathbf{x}_L, \boldsymbol{\nu}_L) \approx \mathbf{g}(\hat{\mathbf{x}}_L, 0) + \hat{\mathbf{g}}'_{\mathbf{x}_L} \cdot (\mathbf{x}_L - \hat{\mathbf{x}}_L) + \hat{\mathbf{g}}'_{\boldsymbol{\nu}_L} \boldsymbol{\nu}_L, \\
&p(\mathbf{x}_l | \mathbf{x}_{l-1}; \boldsymbol{\theta}_{\mathbf{W}_l}) = \mathcal{N}\left(\hat{\mathbf{x}}_l + \hat{\mathbf{f}}'_{\mathbf{x}_{l-1}} \cdot (\mathbf{x}_{l-1} - \hat{\mathbf{x}}_{l-1}), \hat{\mathbf{f}}'_{\mathbf{W}_l} P_{\mathbf{W}_l} \hat{\mathbf{f}}'^{\top}_{\mathbf{W}_l}\right), \qquad (3) \\
&p(\mathbf{y} | \mathbf{x}_L) = \mathcal{N}\left(\mathbf{g}(\hat{\mathbf{x}}_L, 0) + \hat{\mathbf{g}}'_{\mathbf{x}_L} \cdot (\mathbf{x}_L - \hat{\mathbf{x}}_L), \mathbf{g}'_{\boldsymbol{\nu}_L} \mathbf{g}'^{\top}_{\boldsymbol{\nu}_L}\right). \qquad (4)
\end{aligned}
$$

In the above equations, the transformation $\mathbf{f}_l$, mapping post-activation $\mathbf{x}_{l-1}$ to post-activation $\mathbf{x}_l$, is approximated using a first-order Taylor expansion centered around $\mathbf{x}_{l-1} = \hat{\mathbf{x}}_{l-1}$ and $\hat{\mathbf{W}}_l$. Specifically, $\hat{\mathbf{x}}_l = \mathbf{f}_l(\hat{\mathbf{x}}_{l-1}, \hat{\mathbf{W}}_l)$ represents the mean transformation, and $\hat{\mathbf{f}}'_{\mathbf{x}_{l-1}}$ and $\hat{\mathbf{f}}'_{\mathbf{W}_l}$ are the gradients of $\mathbf{f}_l$ with respect to $\mathbf{x}_{l-1}$ and $\mathbf{W}_l$, computed at this point. Similarly, the transformation $\mathbf{g}$ that maps post-activation $\mathbf{x}_L$ and the multivariate standard Gaussian noise generator $\boldsymbol{\nu}_L$ to the observation $\mathbf{y}$ is also approximated using a first-order Taylor expansion centered around $\mathbf{x}_L = \hat{\mathbf{x}}_L$ and $\boldsymbol{\nu}_L = 0$.

Learning a Gaussian linear Bayesian neural network involves inferring the posterior distributions for post-activations and weights through variational or MCMC methods. This process includes a forward "filtering" pass, updating Bayesian beliefs $\alpha(\mathbf{x}_l; \boldsymbol{\theta}_l) \overset{\text{def}}{=} p(\mathbf{x}_l | \mathbf{x}_0 = \mathbf{x})$ from the input to higher-level post-activations and output, and a backward "smoothing" pass, refining the beliefs $\gamma(\mathbf{x}_l; \boldsymbol{\theta}_{l|L}) \overset{\text{def}}{=} p(\mathbf{x}_l | \mathbf{x}_0, \mathbf{y})$, and $q(\mathbf{W}_l; \boldsymbol{\theta}_{\mathbf{W}_l})$ using information from the output and higher-level post-activations, where $\boldsymbol{\theta}_l$, $\boldsymbol{\theta}_{l|L}$, and $\boldsymbol{\theta}_{\mathbf{W}_l}$ are variational parameters. The Kalman filter/smoother, a deterministic algorithm, computes the mean and variance of post-activations and weights as follows:

$$
\begin{aligned}
&\alpha(\mathbf{x}_l) = \mathcal{N}(\hat{\mathbf{x}}_l, P_l), \text{with } P_l = \hat{\mathbf{f}}'_{\mathbf{x}_{l-1}} P_{l-1} \hat{\mathbf{f}}'^{\top}_{\mathbf{x}_{l-1}} + \hat{\mathbf{f}}'_{\mathbf{W}_l} P_{\mathbf{W}_l} \hat{\mathbf{f}}'^{\top}_{\mathbf{W}_l}, \qquad (5) \\
&\gamma(\mathbf{x}_l) = \mathcal{N}\left(\hat{\mathbf{x}}_{l|L}, P_{l|L}\right), \text{ with } \hat{\mathbf{x}}_{L|L} = \hat{\mathbf{x}}_L + K_L(\mathbf{y} - \hat{\mathbf{y}}), P_{L|L} = P_L - K_L S_L K_L, \qquad (6) \\
&\hat{\mathbf{x}}_{l|L} = \hat{\mathbf{x}}_l + G_l\left(\hat{\mathbf{x}}_{l+1|L} - \hat{\mathbf{x}}_{l+1}\right), P_{l|L} = P_l - G_l\left(P_{l+1|L} - P_{l+1}\right) G_l^{\top}, \\
&\hat{\mathbf{W}}_{l|L} = \hat{\mathbf{W}}_l + G_{\mathbf{W}_l}\left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right), P_{\mathbf{W}_{l|L}} = P_{\mathbf{W}_l} - G_{\mathbf{W}_l}\left(P_{l|L} - P_l\right) G_{\mathbf{W}_l}^{\top},
\end{aligned}
$$

where $S_L = \hat{\mathbf{g}}'_{\mathbf{x}_L} P_L \hat{\mathbf{g}}'^{\top}_{\mathbf{x}_L} + \hat{\mathbf{g}}'_{\boldsymbol{\nu}_L} \hat{\mathbf{g}}'^{\top}_{\boldsymbol{\nu}_L}$ represents the observation covariance, $K_L = P_L \mathbf{g}'^{\top}_{\mathbf{x}_L} S_L^{-1}$ is the Kalman gain from the observation, $G_l = P_l \hat{\mathbf{f}}'^{\top}_{\mathbf{x}_l} P^{-1}_{l+1|L}$ and $G_{\mathbf{W}_l} = P_{\mathbf{W}_l} \hat{\mathbf{f}}'^{\top}_{\mathbf{W}_l} P^{-1}_{l|L}$ denote the smoothing gains in backpropagation, and $\hat{\mathbf{W}}_{l|L}$, $P_{\mathbf{W}_{l|L}}$, $\hat{\mathbf{W}}_l$, and $P_{\mathbf{W}_l}$ are the posterior and prior parameters of the weight $\mathbf{W}_l$.

3 Markov Random Field Specified by Bayesian Neural Network

In this section, we consider tensor transformations within a Bayesian neural network as a stochastic process. We then formulate the forward propagation of tensor distributions and the

backpropagation of their sensitivities as a form of belief propagation. We establish a connection between the stochastic perspective on tensor transformations and the deterministic perspective on tensor distribution evolution. Lastly, we present an algorithm for converting a tensor layer into a stochastic layer.

3.1 BACKPROPAGATION THROUGH STOCHASTIC LAYERS

We will explore the forward propagation of tensor distribution parameters in a Bayesian neural network and the subsequent backward propagation of the loss gradient relative to these parameters within the network's computational graph. A theorem presented here establishes a profound mathematical link between two pivotal concepts: backpropagation, essential in neural network training, and belief propagation, commonly employed in probabilistic graphical models. This theorem sheds light on the flow of information through a Bayesian neural network during both forward and backward passes.

In its essential form, the filtering distribution $\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l) = p(\mathbf{x}_l|\mathbf{x}_0 = \mathbf{x})$ is parameterized by $\boldsymbol{\theta}_l$, while the smoothing distribution is $\gamma(\mathbf{x}_l) = p(\mathbf{x}_l|\mathbf{x}_0, \mathbf{y})$. For the exponential family, $\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l) = \exp(-\boldsymbol{\theta}_l \cdot \mathbf{T}(\mathbf{x}_l) - A(\boldsymbol{\theta}_l))$, with canonical parameters $\boldsymbol{\theta}_l$, feature statistics $\mathbf{T}(\mathbf{x}_l)$, and mean parameters $\boldsymbol{\mu}_l = \mathbf{E}_{\alpha(\mathbf{x}_l)}\mathbf{T}(\mathbf{x}_l)$ and $\boldsymbol{\mu}_{l|L} = \mathbf{E}_{\gamma(\mathbf{x}_l)}\mathbf{T}(\mathbf{x}_l)$. In a Gaussian linear model approximation of a Bayesian neural network, post-activation mean and variance are $\hat{\mathbf{x}}_l$, P_l, $\hat{\mathbf{x}}_{l|L}$, and $P_{l|L}$, with canonical parameters $\boldsymbol{\eta}_l = P_l^{-1}\mathbf{x}_l$, $\boldsymbol{\Lambda}_l = -.5 \cdot P_l^{-1}$ for filtering, and $\boldsymbol{\eta}_{l|L}$, $\boldsymbol{\Lambda}_{l|L}$ for smoothing.

Theorem 1. ***Equivalence between Backpropagation and Belief Propagation in a Bayesian Neural Network.***

(1) ***Gradient of Loss with Respect to Parameters:*** *In its most general form, the gradient of loss with respect to the post-activation filtering distribution parameters $\boldsymbol{\theta}_l$ and the variational weights distribution parameters $\boldsymbol{\theta}_{\mathbf{W}_l}$ can be expressed as:*

$$\nabla_{\boldsymbol{\theta}_l} J = -\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{\boldsymbol{\theta}_l} \log \alpha(\mathbf{x}_l;\boldsymbol{\theta}_l), \qquad \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} J = -\mathbf{E}_{\gamma(\mathbf{x}_{l-1},\mathbf{x}_l)} \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} \log p(\mathbf{x}_l|\mathbf{x}_{l-1};\boldsymbol{\theta}_{\mathbf{W}_l}).$$

(2) ***Error Gradient for Canonical Parameters:*** *When filtering distribution is in the exponential family, error gradient for the canonical parameters is the difference in the mean parameters between smoothing and filtering distributions. Backpropagation mirrors backward belief propagation:*

$$\nabla_{\boldsymbol{\theta}_l} J = \boldsymbol{\mu}_{l|L} - \boldsymbol{\mu}_l, \quad \nabla_{\boldsymbol{\theta}_l} J = \nabla_{\boldsymbol{\theta}_l}\boldsymbol{\mu}_l \, \nabla_{\boldsymbol{\mu}_l}\boldsymbol{\mu}_{l+1} \, \nabla_{\boldsymbol{\mu}_{l+1}}\boldsymbol{\theta}_{l+1} \, \nabla_{\boldsymbol{\theta}_{l+1}} J, \quad \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} J = \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}}\boldsymbol{\mu}_l \, \nabla_{\boldsymbol{\mu}_l}\boldsymbol{\theta}_l \, \nabla_{\boldsymbol{\theta}_l} J.$$

(3) ***Error Gradient for Post-Activation Mean and Variance:*** *In a Gaussian linear Bayesian neural network, the error gradient for post-activation mean and variance is related to the difference between smoothing and filtering mean and variance in the corresponding Gaussian linear dynamics. Gradient backpropagation parallels belief backward propagation induced by smoothing gain G_l and $G_{\mathbf{W}_l}$:*

$$\nabla_{\hat{\mathbf{x}}_l} J = -P_l^{-1}\left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right), \nabla_{P_l} J = -.5P_l^{-1}\left(P_{l|L} - P_l + (\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l)(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l)^\top\right) P_l^{-1}, \quad (7)$$

$$\nabla_{\hat{\mathbf{x}}_l} J = P_l^{-1} G_l P_{l+1} \cdot \left(\nabla_{\hat{\mathbf{x}}_{l+1}} J\right), \nabla_{P_l} J = P_l^{-1} G_l P_{l+1} \cdot \left(\nabla_{P_{l+1}} J\right) \cdot P_{l+1} G_l P_l^{-1},$$

$$\nabla_{\hat{\mathbf{W}}_l} J = P_{\mathbf{W}_l}^{-1} G_{\mathbf{W}_l} P_l \cdot \left(\nabla_{\hat{\mathbf{x}}_l} J\right), \nabla_{\boldsymbol{\sigma}^2_{\mathbf{W}_l}} J = \operatorname{diag}\left(G_{\mathbf{W}_l} P_l \left(\nabla_{P_l} J\right) P_l G_{\mathbf{W}_l}\right) / \boldsymbol{\sigma}^4_{\mathbf{W}_l}. \quad (8)$$

The error gradient for the canonical parameters of post-activation filter distributions in a Gaussian linear Bayesian neural network is the difference in the first and second moments between smoothing and filtering distributions in the corresponding Gaussian linear dynamics. Gradient backpropagation again parallels belief backward propagation induced by smoothing gain:

$$\nabla_{\boldsymbol{\eta}_l} J = -\left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right), \qquad \nabla_{\boldsymbol{\Lambda}_l} J = -\left(P_{l|L} + \hat{\mathbf{x}}_{l|L}\hat{\mathbf{x}}_{l|L}^\top - P_l - \hat{\mathbf{x}}_l\hat{\mathbf{x}}_l^\top\right),$$

$$\nabla_{\boldsymbol{\eta}_l} J = G_l \cdot \nabla_{\boldsymbol{\eta}_{l+1}} J, \qquad \nabla_{\boldsymbol{\Lambda}_l} J = G_l \nabla_{\boldsymbol{\Lambda}_{l+1}} J \, G_l^\top - \hat{\mathbf{x}}_l \nabla_{\boldsymbol{\eta}_{l+1}}^\top J \, G_l^\top - G_l \nabla_{\boldsymbol{\eta}_{l+1}} J \, \hat{\mathbf{x}}_l^\top.$$

In the above, (1) is established through a common technique involving the manipulation of logarithms within gradients. It allows us to compute the gradient of loss with respect to the parameters of the filtering distribution.

$$\nabla_{\boldsymbol{\theta}_l} \log p(y|x_0) = \frac{\nabla_{\boldsymbol{\theta}_l} \int dx_l p(y|x_l, x_0) p(x_l|x_0;\theta_l)}{p(y|x_0)} = \int dx_l \frac{p(y|x_l, x_0) p(x_l|x_0)}{p(y|x_0)} \nabla_{\boldsymbol{\theta}_l} \log p(x_l|x_0;\theta_l).$$

The exponential family assumption in (2) generally holds as long as the joint probability density of the random variables is strictly positive, as per the Hammersley-Clifford theorem (Hammersley & Clifford, 1971). This assumption enables general forward-propagation of mean parameters and backpropagation of canonical parameter sensitivities. The gradients of weight variational posterior parameters can be computed using these mean parameters. In (3), sensitivity over mean and variance is converted into canonical parameter sensitivity, backpropagated using the probability kernel, and then reconverted into mean and variance sensitivity, as illustrated by expressions like $P_l^{-1}G_lP_{l+1}\cdot\left(\nabla_{\hat{\mathbf{x}}_{l+1}}J\right)$ and $P_l^{-1}G_lP_{l+1}\cdot\left(\nabla_{P_{l+1}}J\right)\cdot P_{l+1}G_lP_l^{-1}$. Both forward and backward propagation processes involve the exchange of "innovations" between target and current distribution mean parameters, represented as cross-entropy loss gradients in terms of the probability kernels. Equation 8 is presented in a differential form due to its dependence on weight parameterization. A detailed derivation of the natural gradient for Gaussian linear Bayesian neural networks is provided in Section D.

The proof is presented in Section A. In Section C, we derive the backpropagation formula for computing the gradient of the loss with respect to the filter distribution parameters in both a hidden Markov model and a Gaussian linear model. This derivation allows us to draw comparisons between backpropagation and backward message passing. Consequently, Theorem 1 serves a dual role: it enables variational inferences within the deep learning framework for probabilistic graphical models and contributes to the advancement of probabilistic graphical model techniques for Bayesian deep learning.

3.2 Duality between Bayesian Neural Network and Probabilistic Graphical Models

A Bayesian neural network represents a probabilistic ensemble of neural networks operating within a common computational graph. These networks guide tensor "particles" through this graph, adjusting their paths based on gradient information to minimize their respective losses. As they learn and adapt, the probability distribution describing their trajectories evolves. The subsequent theorem establishes a connection between the probabilistic perspective of tensor flow and the deterministic perspective of tensor distribution evolution, focusing specifically on the first and second moments of the gradient.

Theorem 2. ***Relationship between the Langevin and the Fokker-Planck Dynamics of a Bayesian Neural Network.***

(1) For post-activations distributed as per an exponential family, the error gradient relative to post-activation equals the potential energy gradient, resulting from differences in canonical parameters between filtering and smoothing distributions:

$$\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})=\nabla_{\mathbf{x}_l}\log\frac{p(\mathbf{x}_l|\mathbf{y},\mathbf{x}_0=\mathbf{x})}{p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x})}=\left(\nabla_{\mathbf{x}_l}\mathbf{T}(\mathbf{x}_l)\right)^\top\left(\boldsymbol{\theta}_l-\boldsymbol{\theta}_{l|\boldsymbol{L}}\right).$$

This difference in canonical parameters defines the orthogonal projection from the Jacobian of the sufficient statistics to the error gradient relative to post-activation. Here, $\bullet^+$ is the pseudo-inverse.

$$\boldsymbol{\theta}_l-\boldsymbol{\theta}_{l|\boldsymbol{L}}=\left(\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}\mathbf{T}(\mathbf{x}_l)\right)^+\left(\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})\right).$$

(2) In Gaussian-linear Bayesian neural networks, drift and diffusion processes, defined by the error gradient and Hessian, guide post-activations toward the variational posterior of Gaussian linear dynamics as follows:

$$\begin{aligned}
&\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})=P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l)-P_{l|L}^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}),\\
&\nabla_{\mathbf{x}_l\mathbf{x}_l^\top}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})=P_l^{-1}-P_{l|L}^{-1},\\
&\nabla_{\hat{\mathbf{x}}_l}\log p(\mathbf{y}|\mathbf{x}_0=\mathbf{x})=\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x}),\\
&\nabla_{P_l}\log p(\mathbf{y}|\mathbf{x}_0=\mathbf{x})=\nabla_{\mathbf{x}_l\mathbf{x}_l^\top}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})+\mathbf{E}_{\gamma(\mathbf{x}_l)}\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})\right)\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})\right)^\top.
\end{aligned}$$

(3) When the weight variance vanishes, making the state transition deterministic with $p(\mathbf{x}_{l+1}|\mathbf{x}_l)=\delta(\mathbf{x}_{l+1}-f(\mathbf{x}_l))$, Bayesian back propagation degenerates into non-Bayesian back propagation:

$$\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l)=\nabla_{\mathbf{x}_l}\mathbf{x}_{l+1}\nabla_{\mathbf{x}_{l+1}}\log p(\mathbf{y}|\mathbf{x}_{l+1}).$$

The proof is provided in Section B. In (1), we compute $\nabla_{\mathbf{x}_l} \log p(\mathbf{y}|\mathbf{x}_l, \mathbf{x}_0 = \mathbf{x})$ by introducing the filter probability distribution $\alpha(\mathbf{x}_l; \boldsymbol{\theta}_l)$. In (2), both $\nabla_{\mathbf{x}_l} J$ and $\nabla_{\mathbf{x}_l \mathbf{x}_l^\top} J$ describe the drift and diffusion of $\mathbf{x}_l$ toward its posterior distribution. In (3), we calculate $\nabla_{\mathbf{x}_l} \log p(\mathbf{y}|\mathbf{x}_l)$ by introducing the Dirac delta distribution for $\mathbf{x}_l$, which transfers the gradient onto its parameters, or by setting the prior variance P_l of $\mathbf{x}_l$ to 0.

This theorem unites probabilistic tensor flow with deterministic tensor distribution evolution, showing that Langevin dynamics in Bayesian neural networks, particularly weight updates, are equivalent to the Fokker-Planck equation governing weight distribution evolution. This insight holds practical value for Bayesian deep learning, offering potential for novel optimization algorithms for improved generalization and convergence. It also deepens our understanding of how deep neural networks explore weight spaces during training.

As an illustrative example, approximating batch normalization as $\left(\frac{\mathbf{w}^\top \mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + V\right) U$ aligns with variational Bayesian learning using stochastic scale-bias distributions $q(U)$ and $q(V)$, with a uniform prior on $\mathbf{w}$ ensuring $\|\mathbf{w}\| = 1$ (Shekhovtsov & Flach, 2019). This approach hints at using Bayesian layers to streamline neural network designs by reducing dependency on normalization layers (Zagoruyko & Komodakis, 2017; Brock et al., 2021). Additionally, gated recurrent unit-based networks can be conceptualized as hierarchical graphical models, incorporating binary features for input selection (Garner & Tong, 2020), paving the way for novel Bayesian recurrent neural network designs and interpretations. Furthermore, treating Bayesian neural networks as Gaussian Markov random fields offers new algorithmic possibilities and insights in Bayesian optimization, architecture design, and generalization (Snoeyink & Picheny, 2012; Arora et al., 2019a;b).

3.3 A Deterministic BNN Backpropagation Algorithm

Given the dual relationship between forward-backward propagation and belief propagation, we propose the following algorithm for propagating element-wise means and variances of hidden features, along with their sensitivities, while simultaneously computing the gradient of the loss with respect to the weight parameters.

Algorithm 1: Training BNN with backpropagation

Input: Gaussian linear Bayesian neural network (Eqs. 3, 4).

Output: mean parameters $\hat{\mathbf{x}}_l$ and $\mathbf{P}_l$, their sensitivities $\nabla_{\hat{\mathbf{x}}_l} J$ and $\nabla_{\mathbf{P}_l} J$, gradient over weight distribution parameters $\nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} J$.

Forward propagation: For $l = 1, \ldots, L$, $\hat{\mathbf{x}}_l = \mathbf{f}(\hat{\mathbf{x}}_{l-1}, \boldsymbol{\theta}_{\mathbf{W}_l}, \boldsymbol{\omega}_l = 0)$, $\mathbf{P}_l = \hat{\mathbf{f}}'^2_{\mathbf{x}_{l-1}} \cdot \mathbf{P}_{l-1} + \hat{\mathbf{f}}'^2_{\boldsymbol{\omega}_l} \cdot \mathbf{1}$.

Backpropagation: For $l = L, ..., 1$, $\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l = G_l(\hat{\mathbf{x}}_{l+1|L} - \hat{\mathbf{x}}_{l+1})$, $\mathbf{P}_{l|L} - \mathbf{P}_l = G_l^2(\mathbf{P}_{l+1|L} - \mathbf{P}_{l+1})$.

Gradient: Eq. 8.

$\hat{\mathbf{x}}_{L|L} = \hat{\mathbf{x}}_L + P_L\left(\hat{\mathbf{g}}'^\top_{\mathbf{x}_L} \cdot \frac{\mathbf{y} - \mathbf{g}(\hat{\mathbf{x}}_L, 0)}{S_L}\right)$, $\mathbf{P}_{L|L} = \mathbf{P}_L - \mathbf{P}_L^2\left(\left(\hat{\mathbf{g}}'^2_{\mathbf{x}_L}\right)^\top \cdot \frac{1}{\mathbf{S}_L}\right)$, $\mathbf{S}_L = \hat{\mathbf{g}}'^2_{\mathbf{x}_L} \cdot \mathbf{P}_L + \hat{\mathbf{g}}'^2_{\boldsymbol{\nu}_L} \cdot \mathbf{1}$, and $\mathbf{G}_l = \hat{\mathbf{f}}'^\top_{\mathbf{x}_l}\left(\mathbf{P}_l \cdot \mathbf{P}_{l+1}^{-\top}\right)$. "$\cdot$" is matrix multiplication. $\mathbf{P}$ and $\bullet^2$ are element-wise. $\mathbf{f}$, $\hat{\mathbf{x}}_l$, $\mathbf{P}_l$, $\hat{\mathbf{f}}'_{\mathbf{x}_l}$, $\hat{\mathbf{f}}'_{\boldsymbol{\omega}_l}$, $\hat{\mathbf{g}}'_{\mathbf{x}_L}$, $\hat{\mathbf{g}}'_{\boldsymbol{\nu}_L}$ are defined in Eqs. 3 and 4. $\mathbf{1}$ is a 1-vector.

In a computational graph defined as $\mathbf{x}_0 = \mathbf{x}$, $\mathbf{a}_l = \mathbf{W}_l \cdot \mathbf{x}_{l-1}$, and $\mathbf{x}_l = \mathbf{f}_l(\mathbf{a}_l)$ for layers $l = 1, \ldots, L$, along with a mean-field Gaussian variational posterior for synaptic weights $\mathbf{W}_l \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{W}_l}, \boldsymbol{\sigma}^2_{\mathbf{W}_l})$, implementing Algorithm 1 for Bayesian inference under these conditions is straightforward. The process involves propagating the activation mean ($\hat{\mathbf{x}}_{l-1}$) through the layer transformation ($\mathbf{f}_l$) once to compute $\hat{\mathbf{x}}_l$. Subsequently, activation variance (P_{l-1}) is propagated through these layer functions twice: first as $\mathbf{f}_l(\boldsymbol{\mu}^2_{\mathbf{W}_l} \cdot P_{l-1})$ and then as $\mathbf{f}_l(\boldsymbol{\sigma}^2_{\mathbf{W}_l} \cdot \hat{\mathbf{x}}^2_{l-1})$. The final variance ($\mathbf{P}_l$) is computed using element-wise multiplications and a smooth max (log-sum-exp) operation within the max-pooling process.

This algorithm is highly memory-efficient, involving element-wise gradient computations ($\hat{\mathbf{f}}'_{\mathbf{x}_l}$, $\hat{\mathbf{f}}'_{\boldsymbol{\omega}_l}$, $\hat{\mathbf{g}}'_{\mathbf{x}_L}$, $\hat{\mathbf{g}}'_{\boldsymbol{\nu}_L}$), as well as matrix-vector multiplications with a smoothing gain (G_l) to facilitate gradient descent with second-order information. In addition to updating synaptic weights, this algorithm simultaneously manages and updates weight variance on an element-wise basis. The power of automatic differentiation comes into play, streamlining the entire process. All that's required is specifying $\mathbf{P}_l$, a task that can also be automated to further enhance efficiency.

4 EXPERIMENTS

This section highlights belief propagation's effectiveness in training Bayesian neural networks for image classification, showing their competitive performance without normalization. We also evaluate UCI regression datasets and use visualization to understand model behavior. Our code demonstrates symbolic layers for probability distributions and gradients, revealing equivalence between Bayesian neural networks and graphical models. This expands auto-differentiation in TensorFlow, PyTorch, and JAX to support graphical models and variational inference techniques.

4.1 CLASSIFICATION

We demonstrate the effectiveness of the belief propagation algorithm in training various neural network architectures for image classification, considering them as Bayesian neural networks. Particularly, we show that incorporating natural gradient descent significantly improves generalization, eliminating the need for normalization techniques to stabilize and expedite training. Our experiments are designed for efficient one-day training cycles using Google Colab V100/A100. We utilize datasets including CIFAR 10/100 (Krizhevsky, 2009) and Tiny ImageNet (Le & Yang, 2015), along with architectural choices such as DenseNet-BC (Huang et al., 2017), ResNet (He et al., 2016), WideResNet-28-10 (Zagoruyko & Komodakis, 2016), EfficientNet B0 (Tan & Le, 2019), and MLP Mixer-S (Tolstikhin et al., 2021). Our training procedure adopts state-of-the-art practices, featuring a cosine annealing learning rate schedule (Loshchilov & Hutter, 2016) and image augmentation methods (Yun et al., 2019; Zhang et al., 2018b; Cubuk et al., 2020). We compare different configurations, including SGD with normalization layers (`vanilla NN`), Bayes-by-backprop (Blundell et al., 2015) with normalization layers (`reparam.`), Bayes-by-backprop with multiple neural networks per mini-batch without batch normalization (`reparam./NF+NG+ensemble`), belief propagation according to Algorithm 1 without batch normalization (`BP/NF`), and belief propagation with natural gradient without normalization (`BP/NF+NG`).

Table 1 presents the test errors. We observe significant performance improvements with Bayesian learning, particularly in less-regularized architectures. Natural gradient descent also enhances performance (`BP/NF` vs. `BP/NF +NG`). Additionally, Bayes-by-backprop, combined with an ensemble of neural networks and natural gradient descent, achieves competitive results, highlighting the variational aspect of normalization. Notably, we find that Bayesian learning without weight parameter sampling (Algorithm 1) exhibits faster convergence compared to methods involving weight sampling.

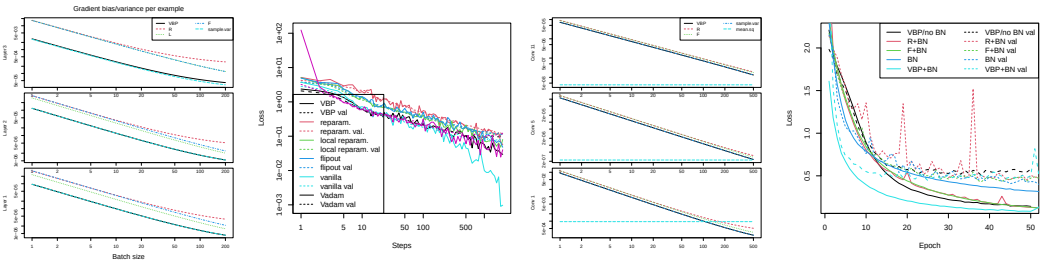
To assess the impact of weight randomness and sample randomness on Bayesian neural network (BNN) convergence and generalization, we analyzed empirical gradient variance with varying batch sizes and the evolution of training/validation loss over epochs. We conducted these analyses using MNIST and CIFAR 10 datasets, training a BNN with densely connected layers and a VGG16 model following the approach of (Wen et al., 2018). Our comparisons included our algorithm (`BP`) against reparameterization (`R`) (Blundell et al., 2015), local reparameterization (`L`) (Kingma et al., 2015), flipout (`F`) (Wen et al., 2018), and an ideal scenario with noise solely from mini-batching (`vanilla`). We observed that weight-induced noise in BNNs is substantial compared to mini-batch noise, and Bayesian learning with Algorithm 1 consistently achieved faster convergence than weight-sampling and non-Bayesian methods (Fig. 1a,c,b,d).

4.2 REGRESSION

We utilized UCI regression datasets to assess the performance of Bayesian neural network learning algorithms, aligning our experimental setup with the framework outlined in Hernández-Lobato

Table 1: A Bayesian neural network trained with belief propagation and natural gradient descent (`BP/NF + NG`), along with Bayes-by-backprop using an ensemble of neural networks and natural gradient descent (`reparam./NF + NG + ensemble`), achieves competitive image classification performance without normalization on state-of-the-art architectures.

		vanilla NN	reparam.	reparam./NF +NG+ensemble	BP/NF	BP/NF+NG
CIFAR10	DenseNet-BC	4.51	4.75	4.50	4.40	4.35
	ResNet	5.46	5.70	4.75	4.35	5.10
	WRN-28-10	3.50	3.65	3.50	3.40	3.30
	Eff.NetB0	4.40	5.15	4.35	4.10	3.90
	MLP Mixer-S	8.20	9.35	8.55	6.70	4.85
CIFAR100	DenseNet-BC	22.27	23.05	21.05	21.90	20.50
	ResNet	27.22	29.55	28.35	27.10	25.90
	WRN-28-10	18.80	18.95	18.90	18.30	17.70
	Eff.NetB0	20.50	21.75	20.30	20.05	19.30
	MLP Mixer-S	30.60	32.90	32.25	28.35	26.00
Tiny ImageNet	DenseNet-BC	30.10	30.50	30.15	29.95	29.25
	ResNet	31.50	33.00	32.00	31.50	30.70
	WRN-28-10	28.70	29.15	28.75	28.55	27.30
	Eff.NetB0	29.45	30.00	29.25	29.10	28.95
	MLP Mixer-S	35.85	38.45	36.35	34.30	31.10



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

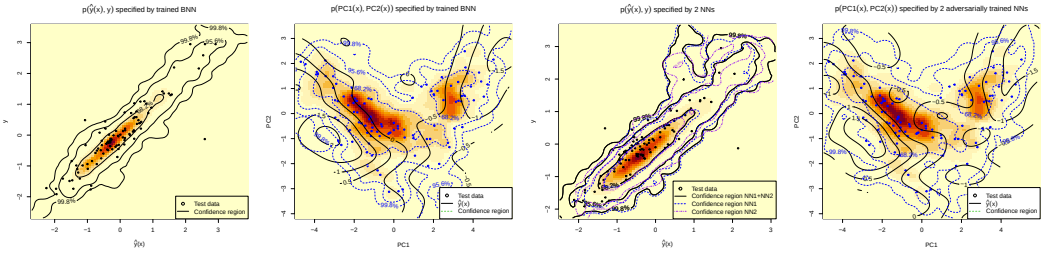
Figure 1: Gradient Variance vs Batch Size for Dense and VGG16 Networks (a and c) and Learning Algorithm Convergence (b and d). Bayesian Neural Formulation Enhances Convergence.

& Adams (2015). Our model (Algorithm 1) is compared against a deep ensemble model (Lakshminarayanan et al., 2017) and the local parameterization algorithm (Kingma et al., 2015). Table 2 presents a comparison of the three approaches based on root mean square error and log probability.

Table 2: Log Probability and RMSE of Belief Propagation, Deep Ensemble, and Local Reparameterization on UCI Data.

	BP		DE		Reparam	
	logprob	rmse	logprob	rmse	logprob	rmse
boston	-2.6 ± 0.4	3.5 ± 1.0	-2.5 ± 0.1	3.3 ± 1.0	-3.2 ± 0.7	3.5 ± 1.1
concrete	-3.2 ± 0.3	6.1 ± 0.8	-3.4 ± 0.1	7.7 ± 0.6	-3.9 ± 0.2	6.8 ± 0.5
energy	-2.2 ± 0.2	2.7 ± 0.4	-2.2 ± 0.1	2.7 ± 0.3	-3.0 ± 0.3	2.9 ± 0.3
kin8nm	1.1 ± 0.0	0.1 ± 0.0	0.4 ± 0.0	0.2 ± 0.0	-2.3 ± 0.5	0.1 ± 0.0
naval	4.0 ± 2.1	0.0 ± 0.0	2.9 ± 0.1	0.0 ± 0.0	4.9 ± 0.1	0.0 ± 0.0
power	-2.8 ± 0.0	4.1 ± 0.2	-3.0 ± 0.0	4.3 ± 0.2	-3.2 ± 0.1	4.2 ± 0.2
protein	-2.9 ± 0.0	4.4 ± 0.1	-3.0 ± 0.0	5.1 ± 0.1	-3.5 ± 0.1	4.5 ± 0.0
wine	-1.0 ± 0.1	0.6 ± 0.0	-0.9 ± 0.1	0.6 ± 0.0	-1.4 ± 0.2	0.6 ± 0.0
yacht	-2.8 ± 0.4	$64 + 19$	$-2.7 + 0.2$	$59 + 17$	$-2.5 + 0.3$	$4.1 + 0.9$

Different from previous works, we also visualize the observation-label distribution of a Bayesian neural network through Monte Carlo simulations and 2D projections (Fig. 2). Starting with slight perturbations to the training data, we employ Hamiltonian Monte Carlo (Neal et al., 2011) to target uncalibrated probabilities (Eq. 2) and monitor convergence with the Gelman-Rubin diagnostic. This produces simulated feature-label patterns representing typical model observations and loss.



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

Figure 2: Joint Distributions from Neural Network Ensembles on Boston Housing Data. Projections: Predicted vs. True Labels (a, c) and Principal Components of Largest Variance (b, d). (a, b) Bayesian NN (Algorithm 1), (c, d) Adversarial NNs.

In Fig. 2 (a) and (c), we project this distribution onto predicted vs. true labels. Ideal models concentrate probability on the diagonal, indicating accurate predictions, especially for unfamiliar cases. Bayesian neural networks trained with Algorithm 1 tightly cluster observations around the diagonal (Fig. 2 (a)), whereas independently trained networks diverge off-diagonal due to limited data (Fig. 2 (c)). Ensembles produce predictions closer to the diagonal. In Fig. 2 (b) and (d), we project observations onto the two principal components, outlining confidence regions and marking predictions. Bayesian neural networks focus observations where predictions incur less loss (Fig. 2 (b)), while non-Bayesian networks have scattered, less smooth expectations. These insights contribute to understanding model behavior, robustness, and susceptibility to adversarial attacks.

5 Related Works

This paper presents a novel perspective on Bayesian neural networks (BNNs) by establishing their equivalence to Markov random fields and linking backpropagation with belief propagation. It distinguishes itself from prior work, which mainly focused on deriving learning algorithms via MCMC and variational inference, using BNNs in NLP and computer vision, and formulating Bayesian learning biases. Bayesian deep learning encompasses both deterministic and MCMC-based methods, including Hamiltonian Monte Carlo (Neal et al., 2011), Stochastic Gradient Langevin Dynamics (Welling & Teh, 2011), and stochastic weight averaging (Maddox et al., 2019). Deterministic techniques like Bayes by backpropagation (Blundell et al., 2015), Laplace approximation (MacKay, 1992; Barber & Bishop, 1998), and natural gradient approximations (Zhang et al., 2018a; Khan et al., 2018) coexist with deep learning-specific methods such as probabilistic backpropagation (Hernández-Lobato & Adams, 2015), deep ensembles (Lakshminarayanan et al., 2017), dropout (Gal & Ghahramani, 2016), and variance-reduction enhancements (Kingma et al., 2015). The revealed equivalence between BNNs and Markov random fields enables the sharing of learning algorithms. BNNs are also versatile, capturing diverse deep learning inductive biases, including adversarial learning (Guo et al., 2017; Ye & Zhu, 2018), semi-supervised learning (Gordon & Hernández-Lobato, 2020), meta-learning (Ravi & Beatson, 2018; Yoon et al., 2018), transfer learning (Maddox et al., 2019), and continual learning (Pan et al., 2020; Daxberger et al., 2021), with robustness against adversarial attacks (Uchendu et al., 2021; Pang et al., 2021). Applications span natural language processing (Shi et al., 2020; Yu et al., 2022), computer vision (Wang et al., 2017), graph learning, time series analysis, and reinforcement learning.

6 Conclusions

Our research illuminates the noteworthy parallels between probabilistic graphical models and Bayesian neural networks, highlighting their synergistic roles in theoretical and practical realms. These similarities foster a reciprocal enhancement of methodologies across both fields. Our empirical validations across diverse scenarios underscore the effectiveness of our approach. This work presents a novel perspective on Bayesian Neural Networks and probabilistic graphical models, fostering advancements in model development and enriching our understanding of their interrelation.

REFERENCES

Sanjeev Arora, Helmut Bolcskei, Lechao Lee, and Meisam Razaviyayn. Neural network architectures and the complexity of learning. *arXiv preprint arXiv:1911.07455*, 2019a.

Sanjeev Arora, Roman Novak, and Lechao Lee. Optimization and generalization of deep learning algorithms with neural tangents. *arXiv preprint arXiv:1912.04759*, 2019b.

David Barber and Christopher M Bishop. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.

Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

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.

Andrew Brock, Soham De, Xiyang Zhuang, and Karen Simonyan. Normalizer free nets (nfnets): High performance large scale image recognition without normalisation. *arXiv preprint arXiv:2102.06171*, 2021.

Ekin D Cubuk, Barret Zoph, and Dandelion Mane. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, 2020.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

Karl Friston. A free energy principle for the brain. *Neural Computation*, 2010.

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.

Crispin W Gardiner et al. *Handbook of stochastic methods*, volume 3. springer Berlin, 1985.

Philip N Garner and Sibo Tong. A bayesian approach to recurrence in neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(8):2527–2537, 2020.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Jonathan Gordon and José Miguel Hernández-Lobato. Combining deep generative and discriminative models for bayesian semi-supervised learning. *Pattern Recognition*, 100:107156, 2020.

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.

J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. *Unpublished Manuscript*, 1971.

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.

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.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pp. 2611–2620. PMLR, 2018.

Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2575–2583, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

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

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Dong Hyun Lee and Yann LeCun. Noise-contrastive priors for energy-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2016.

David JC 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.

Radford M Neal. *BAYESIAN LEARNING FOR NEURAL NETWORKS*. PhD thesis, University of Toronto, 1995.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464, 2020.

Yutian Pang, Sheng Cheng, Jueming Hu, and Yongming Liu. Evaluating the robustness of bayesian neural networks against different types of attacks. *arXiv preprint arXiv:2106.09223*, 2021.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2018.

R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

Havard Rue and Leonhard Held. *Gaussian Markov Random Field Theory and Applications*. CRC press, 2005.

Alexander Shekhovtsov and Boris Flach. Stochastic normalizations as bayesian learning. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part II 14*, pp. 463–479. Springer, 2019.

Wenxian Shi, Hao Zhou, Ning Miao, and Lei Li. Dispersed exponential family mixture vaes for interpretable text generation. In *International Conference on Machine Learning*, pp. 8840–8851. PMLR, 2020.

Jack Snoeyink and Victor Picheny. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1339–1346, 2012.

Yang Song, Zhishen Huang, Aurick Zhou, Serge Belongie, and Mohammad Norouzi. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

David G. Stork, Peter E. Hart, and Richard O. Duda. *Pattern Classification (2nd ed)*. Wiley-Interscience, 2000.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Tijmen Tieleman and Geoffrey Hinton. Contrastive divergence training energy-based models: A unifying view. *Machine Learning*, 2009.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Adaku Uchendu, Daniel Campoy, Christopher Menart, and Alexandra Hildenbrandt. Robustness of bayesian neural networks to white-box adversarial attacks. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 72–80. IEEE, 2021.

Martin J. Wainwright and Michael I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., 2008.

Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Relational deep learning: A deep latent variable model for link prediction. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

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.

Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *International Conference on Learning Representations*, 2018.

John Winn, Christopher M Bishop, and Tommi Jaakkola. Variational message passing. *Journal of Machine Learning Research*, 6(4), 2005.

Nanyang Ye and Zhanxing Zhu. Bayesian adversarial learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6892–6901, 2018.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruiqi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu. Latent diffusion energy-based model for interpretable text modelling. In *International Conference on Machine Learning*, pp. 25702–25720. PMLR, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, and Sanghyuk Chun. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Sergey Zagoruyko and Nikos Komodakis. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861. PMLR, 2018a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2018b.

A PROOF OF THEOREM 1

(1) The gradient of loss with respect to the parameters of the forward/filter probability density can be obtained through the chain rule and the application of the Markov property within the computational graph

$$
\begin{aligned}
&\nabla_{\boldsymbol{\theta}_l} \log \int d\mathbf{x}_l p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x};\boldsymbol{\theta}_l)\cdot p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\\
&= \int d\mathbf{x}_l \nabla_{\boldsymbol{\theta}_l} \underbrace{p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x};\boldsymbol{\theta}_l)}_{\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l)} \cdot p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l) \Big/ \underbrace{\int d\mathbf{x}_l p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x};\boldsymbol{\theta}_l)\cdot p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)}_{p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})}, \text{ chain rule}\\
&= \int d\mathbf{x}_l \nabla_{\boldsymbol{\theta}_l}\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l)\cdot\gamma(\mathbf{x}_l)/\alpha(\mathbf{x}_l) \text{ , because } \frac{p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)}{p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})}=\frac{p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y})}{p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x})}=\frac{\gamma(\mathbf{x}_l)}{\alpha(\mathbf{x}_l)}\overset{\text{def}}{=}\beta(\mathbf{x}_l)\\
&= \int d\mathbf{x}_l \nabla_{\boldsymbol{\theta}_l}\log\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l)\cdot\gamma(\mathbf{x}_l)=\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\boldsymbol{\theta}_l}\log\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l).
\end{aligned}
$$

Similarly, the gradient of the loss concerning weight parameters can be derived through the chain rule and the utilization of the Markov property within the computational graph:

$$
\begin{aligned}
&\nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} \log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})\\
=&\nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} \log \int d\mathbf{x}_{l-1}d\mathbf{x}_l\, p(\mathbf{x}_l|\mathbf{x}_{l-1};\boldsymbol{\theta}_{\mathbf{W}_l})p(\mathbf{x}_{l-1}|\mathbf{x}_0=\mathbf{x})\cdot p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\\
=&\int d\mathbf{x}_{l-1}d\mathbf{x}_l\, \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} p(\mathbf{x}_l|\mathbf{x}_{l-1};\boldsymbol{\theta}_{\mathbf{W}_l})p(\mathbf{x}_{l-1}|\mathbf{x}_0=\mathbf{x})\cdot p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\Big/ p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x}) \text{ , chain rule}\\
&\int d\mathbf{x}_{l-1}d\mathbf{x}_l\, \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} \log p(\mathbf{x}_l|\mathbf{x}_{l-1};\boldsymbol{\theta}_{\mathbf{W}_l})\cdot p(\mathbf{x}_l|\mathbf{x}_{l-1})p(\mathbf{x}_{l-1}|\mathbf{x}_0=\mathbf{x})\cdot p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\Big/ p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})\\
=&\int d\mathbf{x}_{l-1}d\mathbf{x}_l\, \nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} \log p(\mathbf{x}_l|\mathbf{x}_{l-1};\boldsymbol{\theta}_{\mathbf{W}_l})\cdot p(\mathbf{x}_{l-1},\mathbf{x}_l,\mathbf{x}_L|\mathbf{x}_0=\mathbf{x})\Big/ p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})\\
=&\mathbf{E}_{\gamma(\mathbf{x}_{l-1},\mathbf{x}_l)}\nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} \log p(\mathbf{x}_l|\mathbf{x}_{l-1};\boldsymbol{\theta}_{\mathbf{W}_l}), \text{ because } \frac{p(\mathbf{x}_{l-1},\mathbf{x}_l,\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})}{p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})}=p(\mathbf{x}_{l-1},\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y})
\end{aligned}
$$

(2) When the forward/filter distribution belongs to the exponential family, the forward propagation involves iterative updates of the forward mean parameters, denoted as $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_{t+1}(\boldsymbol{\mu}_t)$. Backpropagation, in turn, iteratively updates these mean parameters based on the label information $\mathbf{x}_L = \mathbf{y}$. To compute the gradient of the loss with respect to the forward mean parameters, we substitute the exponential form of the forward/filter probability distribution into $-\nabla_{\boldsymbol{\theta}_l}\log p(\mathbf{y}|\mathbf{x}) = -\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\boldsymbol{\theta}_l}\log\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l)$.

$$
\begin{aligned}
&-\nabla_{\boldsymbol{\theta}_l} J, \text{ where } J\overset{\text{def}}{=}\log p(\mathbf{y}|\mathbf{x})\\
=&-\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\boldsymbol{\theta}_l}\log\alpha(\mathbf{x}_l;\boldsymbol{\theta}_l), \text{ part (1) of the theorem}\\
=&-\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\boldsymbol{\theta}_l}\left(-\boldsymbol{\theta}_l\cdot\mathbf{T}(\mathbf{x}_l)-A(\boldsymbol{\theta}_l)\right)\\
=&\mathbf{E}_{\gamma(\mathbf{x}_l)}\mathbf{T}(\mathbf{x}_l)+\nabla_{\boldsymbol{\theta}_l}A(\boldsymbol{\theta}_l)\\
=&\boldsymbol{\mu}_{l|L}-\boldsymbol{\mu}_l, \text{ definition of } \boldsymbol{\mu}_{l|L} \text{ and the conjugate duality } \boldsymbol{\mu}_l=\nabla_{\boldsymbol{\theta}_l}A(\boldsymbol{\theta}_l)
\end{aligned}
$$

The iterative relationships $\nabla_{\boldsymbol{\theta}_l}J = (\nabla_{\boldsymbol{\theta}_l}\boldsymbol{\mu}_l)\cdot(\nabla_{\boldsymbol{\mu}_l}\boldsymbol{\mu}_{l+1})\cdot\left(\nabla_{\boldsymbol{\mu}_{l+1}}\boldsymbol{\theta}_{l+1}\right)\cdot\left(\nabla_{\boldsymbol{\theta}_{l+1}}J\right)$ and $\nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}} = \left(\nabla_{\boldsymbol{\theta}_{\mathbf{W}_l}}\boldsymbol{\mu}_{l+1}\right)\cdot\left(\nabla_{\boldsymbol{\mu}_{l+1}}\boldsymbol{\theta}_{l+1}\right)\cdot\left(\nabla_{\boldsymbol{\theta}_{l+1}}J\right)$ follow from the chain rule. Note that $\nabla_{\boldsymbol{\theta}_{l+1}}J = \boldsymbol{\mu}_{l+1|L}-\boldsymbol{\mu}_{l+1}$, $\nabla_{\boldsymbol{\mu}_{l+1}}\boldsymbol{\theta}_{l+1} = -\nabla^2_{\boldsymbol{\mu}_{l+1}}H(\boldsymbol{\mu}_{l+1})$ is the Hessian of negative entropy over mean parameters, and $\nabla_{\boldsymbol{\theta}_l}\boldsymbol{\mu}_l = \nabla^2_{\boldsymbol{\theta}_l}A(\boldsymbol{\theta}_l)$ is the Hessian of partition function over canonical parameters.

(3) Substituting the multivariate normal filter distribution of latent features into $-\nabla_{\boldsymbol{\theta}_l} \log p(\mathbf{y}|\mathbf{x}) = -\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{\boldsymbol{\theta}_l} \log \alpha(\mathbf{x}_l; \boldsymbol{\theta}_l)$, we obtain the gradient of loss with respect to the filter mean and variance:

$$
\begin{aligned}
\log \mathcal{N}(\mathbf{x}_l; \hat{\mathbf{x}}_l, P_l) &= -.5 \log \det(2\pi P_l) - .5 \left(\mathbf{x}_l - \hat{\mathbf{x}}_l\right)^\top P_l^{-1} \left(\mathbf{x}_t - \hat{\mathbf{x}}_l\right) \\
\nabla_{P_l} \log \mathcal{N} &= -.5 P_l^{-1} + .5 P_l^{-1} \left(\mathbf{x}_l - \hat{\mathbf{x}}_l\right) \left(\mathbf{x}_l - \hat{\mathbf{x}}_l\right)^\top P_l^{-1} \\
\nabla_{\hat{\mathbf{x}}_l} \log \mathcal{N} &= \cancel{.5 \times 2} P_l^{-1} \left(\mathbf{x}_l - \hat{\mathbf{x}}_l\right) \\
\Rightarrow \mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{P_l} \log \mathcal{N} &= -.5 P_l^{-1} + .5 P_l^{-1} \left(P_{l|L} + \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right) \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right)^\top \right) P_l^{-1} \\
\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{\hat{\mathbf{x}}_l} \log \mathcal{N} &= P_l^{-1} \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right).
\end{aligned}
$$

As a result, we can represent the smoothing mean and variance in terms of the filter mean and variance, along with the gradient of the loss with respect to the filter mean and variance. This formulation allows us to establish an equivalence between the belief propagation algorithm and gradient backpropagation for Gaussian linear models.

$$
\begin{aligned}
\hat{\mathbf{x}}_{l|L} &= \hat{\mathbf{x}}_l + P_l \mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{\hat{\mathbf{x}}_l} \log \mathcal{N} \\
P_{l|L} &= P_l \left(P_l^{-1} + 2\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{P_l} \log \mathcal{N} \right) P_l - \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right) \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right)^\top \\
&= P_l + 2P_l \mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{P_l} \log \mathcal{N} P_l - \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right) \left(\hat{\mathbf{x}}_{l|L} - \hat{\mathbf{x}}_l\right)^\top \\
&= P_l - P_l \left(-2\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{P_l} \log \mathcal{N} + P_l \left(\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{\hat{\mathbf{x}}_l} \log \mathcal{N}\right) \left(\mathbf{E}_{\gamma(\mathbf{x}_l)} \nabla_{\hat{\mathbf{x}}_l} \log \mathcal{N}\right)^\top P_l \right) P_l
\end{aligned}
$$

To identify the chain rule governing the backpropagation of the gradient of loss over forward probability distribution parameters across layers, we initiate by expressing the perturbation in the loss in terms of the perturbation in the cross-entropy between the forward probability distribution and the backward probability distribution at layer l and then propagate this identity backward to layer $l-1$. By leveraging the symbolic relationships between the perturbations of loss, forward parameters at layer l, and forward parameters at layer $l-1$, we can discern how the gradient of loss over these forward parameters propagates backward. In essence, this involves navigating through the following backpropagation process in a differential form.

$$
\begin{aligned}
& d \log p(\mathbf{x}_L = \mathbf{y} | \mathbf{x}_0 = \mathbf{x}) \\
=& \sum_{\mathbf{x}_l} \gamma(\mathbf{x}_l) d \log \alpha(\mathbf{x}_l), \text{ note } \alpha(\mathbf{x}_l) = \sum_{\mathbf{x}_{l-1}} \alpha(\mathbf{x}_{l-1}) p(\mathbf{x}_l | \mathbf{x}_{l-1}) \\
=& \sum_{\mathbf{x}_l} \gamma(\mathbf{x_l}) \frac{\sum_{\mathbf{x}_{l-1}} p(\mathbf{x}_l | \mathbf{x}_{l-1}) d\alpha(\mathbf{x}_{l-1})}{\alpha(\mathbf{x}_l)} \\
=& \sum_{\mathbf{x}_{l-1}} \sum_{\mathbf{x}_l} \gamma(\mathbf{x_l}) \frac{p(\mathbf{x}_l | \mathbf{x}_{l-1}) \alpha(\mathbf{x}_{l-1})}{\alpha(\mathbf{x}_l)} d \log \alpha(\mathbf{x}_{l-1}), \text{ note } \gamma(\mathbf{x_l}) \frac{p(\mathbf{x}_l | \mathbf{x}_{l-1}) \alpha(\mathbf{x}_{l-1})}{\alpha(\mathbf{x}_l)} = \gamma(\mathbf{x}_{l-1}) \\
=& \sum_{\mathbf{x}_{l-1}} \gamma(\mathbf{x}_{l-1}) d \log \alpha(\mathbf{x}_{l-1}).
\end{aligned}
$$

In the third line above, we represent the perturbation of the forward/filter probability distribution at layer l in terms of the perturbation of the forward/filter probability distribution at layer $l-1$. In the fifth line above, we utilize the relationship for forward-propagating the filter probability distribution to backpropagate the smoothing probability distribution at layer l to construct the smoothing probability distribution at layer $l-1$.

In the case of Gaussian linear models, we begin by establishing two key relationships. First, we express the perturbation in the log-likelihood of the forward distribution at layer l in terms of the perturbation in the forward distribution parameters at the same layer. Second, we express the perturbation in the forward distribution parameters at layer l in terms of those at layer $l-1$.

$$
\begin{aligned}
d\log\mathcal{N}(\mathbf{x}_l;\hat{\mathbf{x}}_l,P_l) &= -.5d\log\det(2\pi P_l) - .5d\left[(\mathbf{x}_l-\hat{\mathbf{x}}_l)^\top P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l)\right] \\
&= -.5\text{Tr}(P_l^{-1}dP_l) + .5(\mathbf{x}_l-\hat{\mathbf{x}}_l)^\top P_l^{-1}dP_lP_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l) + d\hat{\mathbf{x}}_l^\top P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l) \\
&= -.5\text{Tr}\left(\left(P_l^{-1} - P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l)(\mathbf{x}_l-\hat{\mathbf{x}}_l)^\top P_l^{-1}\right)dP_l\right) + d\hat{\mathbf{x}}_l^\top P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l) \\
\text{where } dP_l &= d\left(\mathbf{f}'_{\mathbf{x}_{l-1}}P_{l-1}\mathbf{f}'^\top_{\mathbf{x}_{l-1}} + Q_l\right) = \mathbf{f}'_{\mathbf{x}_{l-1}}(dP_{l-1})\mathbf{f}'^\top_{\mathbf{x}_{l-1}} \\
d\hat{\mathbf{x}}_l &= d\mathbf{f}'_{\mathbf{x}_{l-1}}\hat{\mathbf{x}}_{l-1} = \mathbf{f}'_{\mathbf{x}_{l-1}}d\hat{\mathbf{x}}_{l-1}
\end{aligned}
$$

Subsequently, we trace the backpropagation of the perturbation in the forward distribution parameters at layer l to that at layer $l-1$ within the context of the cross-entropy perturbation. This provides us with insights into how the sensitivity of distribution parameters propagates backward.

$$
\begin{aligned}
\Rightarrow &\mathbf{E}_{\gamma(\mathbf{x}_l)}d\log\mathcal{N}(\mathbf{x}_l;\hat{\mathbf{x}}_l,P_l) \\
&= -.5\text{Tr}\left(\left(P_l^{-1} - P_l^{-1}\left(P_{l|L} + (\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l)(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l)^\top\right)P_l^{-1}\right)dP_l\right) + d\hat{\mathbf{x}}_l^\top P_l^{-1}(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l) \\
&= -.5\text{Tr}\left[\underbrace{\left(P_l^{-1} - P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l)(\mathbf{x}_l-\hat{\mathbf{x}}_l)^\top P_l^{-1}\right)}_{\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{P_l}\log\mathcal{N}=\nabla_{P_l}J}\underbrace{\mathbf{f}'_{\mathbf{x}_{l-1}}(dP_{l-1})\mathbf{f}'^\top_{\mathbf{x}_{l-1}}}_{dP_l}\right] + \underbrace{\left(\mathbf{f}'_{\mathbf{x}_{l-1}}d\hat{\mathbf{x}}_{l-1}\right)^\top}_{d\hat{\mathbf{x}}_l^\top}\underbrace{P_l^{-1}(\mathbf{x}_l-\hat{\mathbf{x}}_l)}_{\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\hat{\mathbf{x}}_l}\log\mathcal{N}=\nabla_{\hat{\mathbf{x}}_l}J} \\
&= -.5\text{Tr}\left\{(dP_{l-1})\left[\mathbf{f}'^\top_{\mathbf{x}_{l-1}}(\nabla_{P_l}J)\mathbf{f}'_{\mathbf{x}_{l-1}}\right]\right\} + \text{Tr}\left\{(d\hat{\mathbf{x}}_{l-1})^\top\left[\mathbf{f}'^\top_{\mathbf{x}_{l-1}}(\nabla_{\hat{\mathbf{x}}_l}J)\right]\right\}
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow &\nabla_{\hat{\mathbf{x}}_{l-1}}J = \mathbf{f}'^\top_{\mathbf{x}_{l-1}}(\nabla_{\hat{\mathbf{x}}_l}J) \\
&\nabla_{P_{l-1}}J = \mathbf{f}'^\top_{\mathbf{x}_{l-1}}(\nabla_{P_l}J)\mathbf{f}'_{\mathbf{x}_{l-1}}
\end{aligned}
$$

B PROOF OF THEOREM 2

(1) To find the gradient of the log-likelihood of the label with respect to the activation output, we leverage the Markov property in the joint probability distribution between input $\mathbf{x}_0$, post-activation $\mathbf{x}_l$, and label $\mathbf{x}_L$.

$$\begin{aligned}
\frac{p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x})p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)}{p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})} &= p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y}) \\
\Rightarrow \log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l) &= \log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y}) - \log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x}) + \log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x}) \\
\Rightarrow \nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l) &= \nabla_{\mathbf{x}_l}\underbrace{\log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y})}_{-\boldsymbol{\theta}_{l|L}^\top T(\mathbf{x}_l)-A(\boldsymbol{\theta}_{l|L})} - \nabla_{\mathbf{x}_l}\underbrace{\log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x})}_{-\boldsymbol{\theta}_l^\top T(\mathbf{x}_l)-A(\boldsymbol{\theta}_l)} + \underbrace{\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_0=\mathbf{x})}_{=0} \\
&= \left(\nabla_{\mathbf{x}_l}T(\mathbf{x}_l)\right)^\top(\boldsymbol{\theta}_l-\boldsymbol{\theta}_{l|L})
\end{aligned}$$

In the above $\nabla_{\mathbf{x}_l}T(\mathbf{x}_l)$ is the Jacobian matrix of sufficient statistics where following convention the rows are index by the elements of $\mathbf{x}_l$ and the columns are index by various statistics in the output of $T(\mathbf{x}_l)$. To express the gradient of loss over the canonical parameters of filter/forward post-activation distribution, we note that $\nabla_{\boldsymbol{\mu}_l}J = \boldsymbol{\theta}_l - \boldsymbol{\theta}_{l|L}$ by the duality of mean parameters and canonical parameters, making Monte Carlo estimation of $\boldsymbol{\theta}_l - \boldsymbol{\theta}_{l|L}$ on the one hand, and apply the chain rule to find $\nabla_{\boldsymbol{\theta}_l}J$ on the other hand. Note that the dimensionality of the canonical/mean parameters and the sufficient statistics are higher than the dimensionality of the random variable, and we need to construct more statistics from the sensitivity of post-activation $\nabla_{\mathbf{x}_l}\log p$ to make the system about $\boldsymbol{\theta}_l - \boldsymbol{\theta}_{\boldsymbol{l}|\boldsymbol{L}}$ determined. In this sense $\left(\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}T(\mathbf{x}_l)\right)^+\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}\log p$ is symbolic, because it doesn't specify how to construct additional statistics.

$$\begin{aligned}
\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x}) = \nabla_{\mathbf{x}_l}\log\frac{p(\mathbf{x}_l|\mathbf{y},\mathbf{x}_0=\mathbf{x})}{p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x})} &= \left(\nabla_{\mathbf{x}_l}T(\mathbf{x}_l)\right)^\top\left(\boldsymbol{\theta}_l-\boldsymbol{\theta}_{\boldsymbol{l}|\boldsymbol{L}}\right). \\
\left(\boldsymbol{\theta}_l-\boldsymbol{\theta}_{\boldsymbol{l}|\boldsymbol{L}}\right) &= \left(\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}T(\mathbf{x}_l)\right)^+\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x}), \text{ where } \bullet^+ \text{ is pseudo-inverse} \\
\nabla_{\boldsymbol{\theta}_l}J = \nabla_{\boldsymbol{\theta}_l}\boldsymbol{\mu}_l\cdot\nabla_{\boldsymbol{\mu}_l}J &= \left(\nabla_{\boldsymbol{\theta}_l}\boldsymbol{\mu}_l\right)\cdot\left(\boldsymbol{\theta}_l-\boldsymbol{\theta}_{\boldsymbol{l}|\boldsymbol{L}}\right) \\
\nabla_{\boldsymbol{\theta}_l}J &= \nabla_{\boldsymbol{\theta}_l}\boldsymbol{\mu}_l\cdot\left(\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}T(\mathbf{x}_l)\right)^+\mathbf{E}_{\mathbf{x}_l}\nabla_{\mathbf{x}_l}\log p(\mathbf{y}|\mathbf{x}_l,\mathbf{x}_0=\mathbf{x})
\end{aligned}$$

(2) The following derivation shows that the gradient of loss over post-activation ensemble mean can be stochastically estimated as the average of the gradient over post-activations from many trajectories.

$$\begin{aligned}
\log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x}) &= -.5\log\det(2\pi P_l) - .5\left(\mathbf{x}-\hat{\mathbf{x}}_l\right)^\top P_l^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}_l\right) \\
\log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y}) &= -.5\log\det(2\pi P_{l|L}) - .5\left(\mathbf{x}-\hat{\mathbf{x}}_{l|L}\right)^\top P_{l|L}^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}_{l|L}\right) \\
\Rightarrow \nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l) &= \nabla_{\mathbf{x}_l}\log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x},\mathbf{x}_L=\mathbf{y}) - \nabla_{\mathbf{x}_l}\log p(\mathbf{x}_l|\mathbf{x}_0=\mathbf{x}) \\
&= -P_{l|L}^{-1}\left(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}\right) + P_l^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}_l\right) \\
\Rightarrow \mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l) &= \cancel{-P_{l|L}^{-1}\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_{l|L}\right)} + P_l^{-1}\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right) = -\nabla_{\hat{\mathbf{x}}_{l|L}}J.
\end{aligned}$$

The following derivation shows that the gradient of loss over post-activation ensemble variance can be stochastically estimated as the average kinetic energy plus the diffusion of the post-activations from many trajectories.

$$\begin{aligned}
&\nabla^2_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l) = -P_{l|L}^{-1}+P_l^{-1} \\
&\mathbf{E}_{\gamma(\mathbf{x}_l)}\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)^\top \\
&= \mathbf{E}_{\gamma(\mathbf{x}_l)}\left\{\left[P_l^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}_l\right)-P_{l|L}^{-1}\left(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}\right)\right]\left[P_l^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}_l\right)-P_{l|L}^{-1}\left(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}\right)\right]^\top\right\} \\
&\mathbf{E}_{\gamma(\mathbf{x}_l)}\left[P_{l|L}^{-1}\left(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}\right)\left(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}\right)^\top P_{l|L}^{-1}\right]=P_{l|L}^{-1}P_{l|L}P_{l|L}^{-1}=P_{l|L}^{-1}
\end{aligned}$$

$$\mathbf{E}_{\gamma(\mathbf{x}_l)}\left[P_l^{-1}(\mathbf{x}-\hat{\mathbf{x}}_l)(\mathbf{x}-\hat{\mathbf{x}}_l)^\top P_l^{-1}\right]=P_l^{-1}\left[P_{l|L}+\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)^\top\right]P_l^{-1}$$

$$-\mathbf{E}_{\gamma(\mathbf{x}_l)}\left[P_{l|L}^{-1}\left(\mathbf{x}_l-\hat{\mathbf{x}}_{l|L}\right)(\mathbf{x}-\hat{\mathbf{x}}_l)^\top P_l^{-1}\right]=-P_{l|L}^{-1}\left[P_{l|L}+\cancelto{=0}{\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_{l|L}\right)\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)^\top}\right]P_l^{-1}=-P_l^{-1}$$

$$\begin{aligned}
\Rightarrow&\mathbf{E}_{\gamma(\mathbf{x}_l)}\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)^\top+\nabla^2_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\\
&=\left\{P_{l|L}^{-1}+P_l^{-1}\left[P_{l|L}+\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)^\top\right]P_l^{-1}-2P_l^{-1}\right\}+\left\{-P_{l|L}^{-1}+P_l^{-1}\right\}\\
&=P_l^{-1}\left[P_{l|L}+\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)^\top\right]P_l^{-1}-P_l^{-1}\\
&=-2\nabla_{P_l}J.
\end{aligned}$$

(3) Taking $\mathbf{x}_l\sim\mathcal{N}(\hat{\mathbf{x}}_l;P_l\to 0)$, the expectation of $\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)$ is

$$\begin{aligned}
&\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\\
=&-\nabla_{\hat{\mathbf{x}}_l}J=\mathbf{f}_{\mathbf{x}_{l+1}}^{'\top}\left(\nabla_{\hat{\mathbf{x}}_{l+1}}J\right)=\mathbf{f}_{\mathbf{x}_{l+1}}^{'\top}\mathbf{f}_{\mathbf{x}_{l+2}}^{'\top}\left(\nabla_{\hat{\mathbf{x}}_{l+2}}J\right)=\cdots=\prod_{l'=l+1}^{L}\mathbf{f}_{\mathbf{x}_{l'}}^{'\top}
\end{aligned}$$

The variance of $\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)$ is

$$\begin{aligned}
&\mathrm{Var}\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)\\
=&\mathbf{E}_{\gamma(\mathbf{x}_l)}\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)\left(\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)^\top-\left(\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)\left(\mathbf{E}_{\gamma(\mathbf{x}_l)}\nabla_{\mathbf{x}_l}\log p(\mathbf{x}_L=\mathbf{y}|\mathbf{x}_l)\right)^\top\\
=&\left\{P_{l|L}^{-1}+P_l^{-1}\left[P_{l|L}+\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)^\top\right]P_l^{-1}-2P_l^{-1}\right\}-P_l^{-1}\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)\left(\hat{\mathbf{x}}_{l|L}-\hat{\mathbf{x}}_l\right)^\top P_l^{-1}\\
=&P_{l|L}^{-1}+P_l^{-1}P_{l|L}P_l^{-1}-2P_l^{-1}=\left(P_{l|L}^{-1}-P_l^{-1}\right)+P_l^{-1}\left(P_{l|L}-P_l\right)P_l^{-1}\to 0.
\end{aligned}$$

It can also be proved based on the property of Dirac's delta function δ to transfer the gradient on itself to its parameters.

C BACKPROPAGATING SMOOTHING DENSITY THROUGH TIME

In this section, we will derive the backpropagation formula for computing the gradient of the loss concerning the filter distribution parameters within the context of a hidden Markov model and a Kalman filter, where exact inference methods are applicable. Our objective is to provide further clarity on the connection between message passing and backpropagation.

C.1 BACKPROPAGATING RNN SMOOTHING DENSITY THROUGH TIME

In the following, we derive the backpropagation algorithm to compute the gradient of loss with respect to the filter probability densities of a Bayesian recurrent neural network modeled as a Gaussian linear process.

We model the Bayesian recurrent neural network (Eqs. 3, 4) as the following Gaussian linear process:

- $\mathbf{x}_t = F_t\mathbf{x}_{t-1} + \boldsymbol{\omega}_t$, where $\boldsymbol{\omega}_t \sim \mathcal{N}(\vec{0}, Q_t)$ is the process noise, $\mathbf{x}_t$ is the latent feature at time t (centered so the mean of $\mathbf{x}_t$ is 0), and F_t represents the dynamics of the recurrent neural network.
- $\mathbf{z}_t = H_t\mathbf{x}_t + \boldsymbol{\nu}_t$, where $\boldsymbol{\nu}_t \sim \mathcal{N}(\vec{0}, R_t)$ is the observation noise, $\mathbf{z}_t$ is the observation, and H_t represents the observation model.

A Kalman filter to forward propagate the information in the observations has a prediction step and an update step:

- prediction: $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t-1}, P_{t|t-1})$, where $\hat{\mathbf{x}}_{t|t-1} = F_t\hat{\mathbf{x}}_{t-1|t-1}$ is the predicted (a priori) state, and $P_{t|t-1} = F_tP_{t-1|t-1}F_t^{\mathrm{T}} + Q_t$ is the predicted (a priori) covariance.
- update: $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t) = \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t}, P_{t|t})$ where the updated (a posterior) state estimate $\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + K_t\boldsymbol{\epsilon}_t$ and the updated (a posterior) covariance $P_{t|t} = P_{t|t-1} - K_tH_tP_{t|t-1}$ are computed from the innovation in the observation $\boldsymbol{\epsilon}_t = \mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}$, innovation covariance $S_t = H_tP_{t|t-1}H_t^\top + R_t$, and the optimal Kalman gain $K_t = P_{t|t-1}H_t^\top S_t^{-1}$.

A Kalman smoother updates smoothing densities backward in time: $p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_T) = \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|T}, P_{t|T})$, where the updated (a posterior) state estimate $\hat{\mathbf{x}}_{t|T} = \hat{x}_{k|k} + G_t\boldsymbol{\delta}_t$ and the updated (a posterior) covariance matrix $P_{t|T} = P_{t|t} + G(P_{t+1|T} - P_{t+1|t})G_t^\top$ are computed from the innovation in later time steps $\boldsymbol{\delta}_t = \left(\hat{\mathbf{x}}_{t+1|T} - \hat{\mathbf{x}}_{t+1|t}\right)$, innovation covariance $P_{t+1|t} = F_{t+1}P_{t|t}F_{t+1}^\top + Q_{t+1}$, and the optimal gain in smoothing $G_t = P_{t|t}F_{t+1}^\top P_{t+1|t}^{-1}$.

Theorem 3. *The gradient of loss over filter mean and variance $-\nabla_{\hat{\mathbf{x}}_{t-1|t-1}, P_{t-1|t-1}} \log p(\mathbf{y}_1, \ldots, \mathbf{y}_T)$ in a Bayesian recurrent neural network modeled as a Gaussian linear process can be recursively computed using the following formulae.*

$$\nabla_{\hat{\mathbf{x}}_{t-1|t-1}} \log p = F_t^\top P_{t|t-1}^{-1} P_{t|t} \left(\nabla_{\hat{\mathbf{x}}_{t|t}} \log p\right) + F_t^\top H_t^\top S_{t|t}^{-1} \left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right) \tag{9}$$

$$\begin{aligned} -2\nabla_{P_{t-1|t-1}} \log p =& F_t^\top P_{t|t-1}^{-1} P_{t|t} \left(\nabla_{P_{t|t}} \log p\right) P_{t|t} P_{t|t-1}^{-1} F_t \\ &+ 2F_t^\top P_{t|t-1}^{-1} P_{t|t} \left(\nabla_{\hat{\mathbf{x}}_{t|t}} \log p\right) \left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1} H_t F_t \\ &+ F_t^\top H_t^\top S_t^{-1} H_t F_t - F_t^\top H_t^\top S_t^{-1} \left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right) \left(\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1} H_t F_t. \end{aligned} \tag{10}$$

Proof. To backpropagate sensitivity through time, we first write out the perturbation of filter density parameters at time t, $dP_{t|t}$ and $d\hat{\mathbf{x}}_{t|t}$ in terms of the perturbation of filter density parameters at time $t-1$, $dP_{t-1|t-1}$ and $d\hat{\mathbf{x}}_{t-1|t-1}$, according to the prediction and update steps of Kalman filter. Next, we write out the perturbation of loss with respect to the perturbation of filter density parameters at time t according to Theorem 1 and transform it into the perturbation of loss with respect to those at time $t-1$. Specifically, we write out $d\log\alpha(\mathbf{x}_t) = p(\mathbf{x}_{t-1}|\mathbf{y}_1, \ldots, \mathbf{y}_t, \mathbf{x}_t) d\log\alpha(\mathbf{x}_{t-1})$, and combine $p(\mathbf{x}_{t-1}|y_1, \ldots, \mathbf{y}_t, \mathbf{x}_t)$ with $\gamma(\mathbf{x}_t)$ to backpropagate the smooth density $\gamma(\mathbf{x}_t)p(\mathbf{x}_{t-1}|\mathbf{y}_1, \ldots, \mathbf{y}_t, \mathbf{x}_t) = \gamma(\mathbf{x}_{t-1})$, where

$\alpha(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$ is the filter probability distribution, and $\gamma(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_T)$ is the smoothing probability distribution. The likelihood $p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t})$ joins the back propagation through $\alpha(\mathbf{x}_{t-1})$.

$$
\begin{aligned}
& d_{\log \alpha(\mathbf{x}_t)} \log p = \int d\mathbf{x}_t \gamma(\mathbf{x}_t) d \log \alpha(\mathbf{x}_t), \text{ by theorem 1} \\
=& \int d\mathbf{x}_t \gamma(\mathbf{x}_t) d \left(\log \int d\mathbf{x}_{t-1} \alpha(\mathbf{x}_{t-1}) p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) - \log p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t-1}) \right) \\
=& \int d\mathbf{x}_t \gamma(\mathbf{x}_t) \frac{\int d\mathbf{x}_{t-1} p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) d\alpha(\mathbf{x}_{t-1})}{\alpha(\mathbf{x}_t) p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t-1})} - \frac{\int d\mathbf{x}_{t-1} d\mathbf{x}_t p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) d\alpha(\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t-1})}, \\
& \text{where } \alpha(x_t) = \textstyle\int d\mathbf{x}_{t-1} p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) \alpha(\mathbf{x}_{t-1}) \Big/ p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t-1}) \\
=& \int d\mathbf{x}_{t-1,t} \left[\gamma(\mathbf{x}_t) \frac{p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) \alpha(\mathbf{x}_{t-1})}{\alpha(\mathbf{x}_t) p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t-1})} - \frac{p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) \alpha(\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1,\ldots,t-1})} \right] d \log \alpha(\mathbf{x}_{t-1}) \\
=& \int d\mathbf{x}_{t-1} \gamma(\mathbf{x}_{t-1}) d \log \alpha(\mathbf{x}_{t-1}) - \int d\mathbf{x}_{t-1} \alpha(\mathbf{x}_{t-1}) d \log \alpha(\mathbf{x}_{t-1}). \\
& \text{because } \tfrac{p(x_t, y_t|x_{t-1}) \alpha(x_{t-1})}{\alpha(x_t) p(y_t|y_{1,\ldots,t-1})} = \cdots = p(x_{t-1}|y_1, \ldots, y_t, x_t) = p(x_{t-1}|y_{1,\ldots,T}, x_t)
\end{aligned}
$$

First, we write out the perturbation of filter density parameters at time t, $dP_{t|t}$ and $d\hat{\mathbf{x}}_{t|t}$, in terms of the perturbation of filter density parameters at time $t-1$, $dP_{t-1|t-1}$ and $d\hat{\mathbf{x}}_{t-1|t-1}$.

$$
\begin{aligned}
& dP_{t|t} \qquad (11) \\
=& dP_{t|t-1} - d \left(\underbrace{P_{t|t-1} H_t^\top \overbrace{\left(H_t P_{t|t-1} H_t^\top + R_t \right)^{-1}}^{S_t^{-1}}}_{K_t} H_t P_{t|t-1} \right) \\
=& d \left(F_t P_{t-1|t-1} F_t^\top + Q_t \right) - d \left(P_{t|t-1} H_t^\top S_t^{-1} H_t P_{t|t-1} \right) \\
=& F_t \left(dP_{t-1|t-1} \right) F_t^\top - \underbrace{F_t \left(dP_{t-1|t-1} \right) F_t^\top}_{dP_{t|t-1}} H_t^\top S_t^{-1} H_t P_{t|t-1} - P_{t|t-1} H_t^\top S_t^{-1} H_t F_t \left(dP_{t-1|t-1} \right) F_t^\top \\
& + P_{t|t-1} H_t^\top S_t^{-1} \overbrace{H_t \underbrace{F_t \left(dP_{t-1|t-1} \right) F_t^\top}_{dP_{t|t-1}} H_t^\top}^{dS_t} S_t^{-1} H_t P_{t|t-1},
\end{aligned}
$$

$$
\begin{aligned}
& d\hat{\mathbf{x}}_{t|t} \qquad (12) \\
=& d \underbrace{F_t \hat{\mathbf{x}}_{t-1|t-1}}_{\hat{\mathbf{x}}_{t|t-1}} + \underbrace{P_{t|t-1} H_t^\top \overbrace{\left(H_t P_{t|t-1} H_t^\top + R_t \right)^{-1}}^{S_t^{-1}}}_{K_t} \left(\mathbf{z}_t - H_t \hat{\mathbf{x}}_{t|t-1} \right) \\
=& F_t \left(d\hat{\mathbf{x}}_{t-1|t-1} \right) - K_t H_t F_t \left(d\hat{\mathbf{x}}_{t-1|t-1} \right) + \underbrace{F_t \left(dP_{t-1|t-1} \right) F_t^\top}_{dP_{t|t-1}} H_t^\top S_t^{-1} \left(\mathbf{z}_t - H_t \hat{\mathbf{x}}_{t|t-1} \right) \\
& - P_{t|t-1} H_t^\top S_t^{-1} \overbrace{H_t \underbrace{F_t \left(dP_{t-1|t-1} \right) F_t^\top}_{dP_{t|t-1}} H_t^\top}^{dS_t} S_t^{-1} \left(\mathbf{z}_t - H_t \hat{\mathbf{x}}_{t|t-1} \right).
\end{aligned}
$$

Next, we transform $\int d\mathbf{x}_t \gamma(\mathbf{x}_t) d \log \alpha(\mathbf{x}_t)$ into the sum of $\int d\mathbf{x}_{t-1} \gamma(\mathbf{x}_{t-1}) d \log \alpha(\mathbf{x}_{t-1}) = \int d\mathbf{x}_{t-1} \left(\int d\mathbf{x}_t \gamma(\mathbf{x}_t) p(\mathbf{x}_{t-1}|\mathbf{y}_1, \ldots, \mathbf{y}_t, \mathbf{x}_t) \right) d \log \alpha(\mathbf{x}_{t-1})$ and $- \int d\mathbf{x}_{t-1} \alpha(\mathbf{x}_{t-1}) d \log \alpha(\mathbf{x}_{t-1})$.

The backpropagation of sensitivity with respect to the filter density parameters at time $t-1$ follows from how $\gamma(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{y}_1, \ldots, \mathbf{y}_T)$ is transformed into $\gamma(\mathbf{x}_{t-1})$.

$$
\begin{aligned}
& d\log\mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_t, P_{t|t}) \\
=& -.5d\log\det(2\pi P_{t|t}) - .5d\left[\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)^\top P_{t|t}^{-1}\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)\right] \\
=& -.5\mathrm{Tr}(P_{t|t}^{-1}dP_{t|t}) + .5\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)^\top P_{t|t}^{-1}dP_{t|t}P_{t|t}^{-1}\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right) + d\hat{\mathbf{x}}_{t|t}^\top P_{t|t}^{-1}\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right) \\
=& -.5\mathrm{Tr}\left(\left(P_{t|t}^{-1} - P_{t|t}^{-1}\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)^\top P_{t|t}^{-1}\right)dP_{t|t}\right) + d\hat{\mathbf{x}}_{t|t}^\top P_{t|t}^{-1}\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right) \\
\Rightarrow& \mathbf{E}_\gamma d\log\mathcal{N} \\
=& -.5\mathrm{Tr}\left(\left(P_{t|t}^{-1} - P_{t|t}^{-1}\left(P_{t|T} + \left(\hat{\mathbf{x}}_{t|T} - \hat{\mathbf{x}}_{t|t}\right)\left(\hat{\mathbf{x}}_{t|T} - \hat{\mathbf{x}}_{t|t}\right)^\top\right)P_{t|t}^{-1}\right)dP_{t|t}\right) + d\hat{\mathbf{x}}_{t|t}^\top P_{t|t}^{-1}\left(\hat{\mathbf{x}}_{t|T} - \hat{\mathbf{x}}_{t|t}\right).
\end{aligned}
$$

Substituting Eqs. 11 and 12 into $\mathbf{E}_\gamma d\log\mathcal{N}$ above, we backpropagate the perturbation $\sum_{t'>t} d\log p(\mathbf{y}_{t'}|\mathbf{y}_1, \ldots, \mathbf{y}_{t'-1})$ in terms of $d\log\alpha(\mathbf{x}_t)$ to the perturbation in terms of $d\log\alpha(\mathbf{x}_{t-1})$ through $\int d\mathbf{x}_{t-1}\left(\int d\mathbf{x}_t \gamma(\mathbf{x}_t) p(\mathbf{x}_{t-1}|\mathbf{y}_1, \ldots, \mathbf{y}_t, \mathbf{x}_t)\right) d\log\alpha(\mathbf{x}_{t-1})$:

$$
\begin{aligned}
& \mathbf{E}_{\gamma(\mathbf{x}_t)}\gamma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}_1, \ldots, \mathbf{y}_T) d\log\mathcal{N}(\mathbf{x}_{t-1}; \hat{\mathbf{x}}_{t-1|t-1}, P_{t-1|t-1}) \qquad (13) \\
=& -.5\mathrm{Tr}\Bigg[\underbrace{\left(P_{t|t}^{-1} - P_{t|t}^{-1}\left(P_{t|T} + \left(\hat{\mathbf{x}}_{t|T} - \hat{\mathbf{x}}_{t|t}\right)\left(\hat{\mathbf{x}}_{t|T} - \hat{\mathbf{x}}_{t|t}\right)^\top\right)P_{t|t}^{-1}\right)}_{\mathbf{E}_{\gamma(\mathbf{x}_t)}\nabla_{P_{t|t}}\log\mathcal{N}} \cdot \\
& \underbrace{\left(F_t\left(dP_{t-1|t-1}\right)F_t^\top - F_t\left(dP_{t-1|t-1}\right)F_t^\top H_t^\top S_t^{-1}H_t P_{t|t-1} - P_{t|t-1}H_t^\top S_t^{-1}H_t F_t(dP_{t-1|t-1})F_t^\top + P_{t|t-1}H_t^\top S_t^{-1}H_t F_t\left(dP_{t-1|t-1}\right)F_t^\top H_t^\top S_t^{-1}H_t P_{t|t-1}\right)}_{dP_{t|t}}\Bigg] \\
& + \underbrace{\left(F_t\left(d\hat{\mathbf{x}}_{t-1|t-1}\right) - K_t H_t F_t\left(d\hat{\mathbf{x}}_{t-1|t-1}\right) + F_t\left(dP_{t-1|t-1}\right)F_t^\top H_t^\top S_t^{-1}\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right) - P_{t|t-1}H_t^\top S_t^{-1}H_t F_t\left(dP_{t-1|t-1}\right)F_t^\top H_t^\top S_t^{-1}\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)\right)^\top}_{d\hat{\mathbf{x}}_{t|t}^\top} \cdot \underbrace{P_{t|t}^{-1}\left(\hat{\mathbf{x}}_{t|T} - \hat{\mathbf{x}}_{t|t}\right)}_{\mathbf{E}_{\gamma(\mathbf{x}_t)}\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}} \\
=& -.5\mathrm{Tr}\Big\{\left(dP_{t-1|t-1}\right)\Big[F_t^\top\left(\nabla_{P_{t|t}}\log\mathcal{N}\right)F_t - F_t^\top H_t^\top \underbrace{S_t^{-1}H_t P_{t|t-1}}_{K_t^\top}\left(\nabla_{P_{t|t}}\log\mathcal{N}\right)F_t - F_t^\top\left(\nabla_{P_{t|t}}\log\mathcal{N}\right)\underbrace{P_{t|t-1}H_t^\top S_t^{-1}}_{K_t^\top}H_t F_t \\
& \qquad + F_t^\top H_t^\top \underbrace{S_t^{-1}H_t P_{t|t-1}}_{K_t^\top}\left(\nabla_{P_{t|t}}\log\mathcal{N}\right)\underbrace{P_{t|t-1}H_t^\top S_t^{-1}}_{K_t^\top}H_t F_t\Big]\Big\} \\
& + \left(d\hat{\mathbf{x}}_{t-1|t-1}\right)^\top\left[F_t^\top\left(\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}\right) - F_t^\top H_t^\top K_t^\top\left(\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}\right)\right] \\
& + \mathrm{Tr}\Big\{\left(dP_{t-1|t-1}\right) \\
& \qquad \left[-F_t^\top\left(\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}\right)\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1}H_t F_t + F_t^\top H_t^\top \underbrace{S_t^{-1}H_t P_{t|t-1}}_{K_t^\top}\left(\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}\right)\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1}H_t F_t\right]\Big\} \\
=& -.5\mathrm{Tr}\Big\{\left(dP_{t-1|t}\right)\left[F_t^\top P_{t|t-1}^{-1}P_{t|t}\left(\nabla_{P_{t|t}}\log\mathcal{N}\right)P_{t|t}P_{t|t-1}^{-1}F_t + 2F_t^\top P_{t|t-1}^{-1}P_{t|t}\left(\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}\right)\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1}H_t F_t\right]\Big\} \\
& + \mathrm{Tr}\Big\{\left(d\hat{\mathbf{x}}_{t-1|t-1}\right)^\top\left[F_t^\top P_{t|t-1}^{-1}P_{t|t}\left(\nabla_{\hat{\mathbf{x}}_{t|t}}\log\mathcal{N}\right)\right]\Big\}
\end{aligned}
$$

Substituting Eqs. 11 and 12 into $-d\log p(\mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = -\int d\mathbf{x}_{t-1}\alpha(\mathbf{x}_{t-1}) d\log\alpha(\mathbf{x}_{t-1})$, we express the information in $\mathbf{y}_t$ in terms of $d\log\alpha(\mathbf{x}_{t-1})$ in the following way.

$$
d\log\mathcal{N}\left(\mathbf{y}_t; H_t\hat{x}_{t|t-1}, \underbrace{H_t P_{t|t-1}H_t^\top + R_t}_{S_t}\right) \qquad (14)
$$

$$= -.5d\left(\log\det\left(2\pi S_t\right) - \left(\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1}\left(\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}\right)\right)$$

$$= -.5\mathrm{Tr}(S_t^{-1}H_tF_tdP_{t-1|t-1}F_t^\top H_t^\top) + .5\left(\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1}H_tF_tdP_{t-1|t-1}F_t^\top H_t^\top S_t^{-1}\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right) + d\hat{x}_{t-1|t-1}^\top F_t^\top H_t^\top S_{t|t}^{-1}\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)$$

$$= -.5\mathrm{Tr}\Big[\left(F_t^\top H_t^\top S_t^{-1}H_tF_t - F_t^\top H_t^\top S_t^{-1}\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)\left(\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}\right)^\top S_t^{-1}H_tF_t\right)dP_{t-1|t-1}\Big] + d\hat{\mathbf{x}}_{t-1|t-1}^\top F_t^\top H_t^\top S_{t|t}^{-1}\left(\mathbf{z}_t - H_t\hat{\mathbf{x}}_{t|t-1}\right)$$

Summing the above two perturbation terms in Eqs. 13 and 14 and gather the multipliers of $dP_{t-1|t-1}$ and $d\hat{\mathbf{x}}_{t-1|t-1}^\top$, we get the backpropagation formulae.

□

As a sanity check, we show that the gradient of loss with respect to filter density parameters computed from backpropagation (Eqs. 9,10) is consistent with the analytical form (Eq. 7).

$$\begin{aligned}
&\nabla_{\hat{x}_{t-1|t-1}}\log\mathcal{N}\\
=&F_t^\top P_{t|t-1}^{-1}P_{t|t}\left(\nabla_{\hat{x}_{t|t}}\log\mathcal{N}\right) + F_t^\top H_t^\top S_{t|t}^{-1}\left(z_t - H_t\hat{x}_{t|t-1}\right)\\
=&F_t^\top P_{t|t-1}^{-1}\cancel{P_{t|t}P_{t|t}^{-1}}\left(\hat{x}_{t|T} - \hat{x}_{t|t}\right) + F_t^\top H_t^\top S_{t|t}^{-1}\left(z_t - H_t\hat{x}_{t|t-1}\right)\\
=&F_t^\top P_{t|t-1}^{-1}\left(\hat{x}_{t|T} - \hat{x}_{t|t-1} - K_t\left(z_t - H_t\hat{x}_{t|t-1}\right)\right) + F_t^\top H_t^\top S_{t|t}^{-1}\left(z_t - H_t\hat{x}_{t|t-1}\right)\\
=&P_{t-1|t-1}^{-1}\underbrace{P_{t-1|t-1}F_t^\top P_{t|t-1}^{-1}}_{G_{t-1}}\left(\hat{x}_{t|T} - \hat{x}_{t|t-1}\right) - \underbrace{F_t^\top P_{t|t}^{-1}K_t}_{F_t^\top H_t^\top S_{t|t}^{-1}}\left(z_t - H_t\hat{x}_{t|t-1}\right) + F_t^\top H_t^\top S_{t|t}^{-1}\left(z_t - H_t\hat{x}_{t|t-1}\right)\\
=&P_{t-1|t-1}^{-1}\left(\hat{x}_{t-1|T} - \hat{x}_{t-1|t-1}\right)
\end{aligned}$$

$$\begin{aligned}
&-2\nabla_{P_{t-1|t-1}}\log\mathcal{N}\\
=&\underbrace{F_t^\top P_{t|t-1}^{-1}P_{t|t}\left(\nabla_{P_{t|t}}\log\mathcal{N}\right)P_{t|t}P_{t|t-1}^{-1}F_t}_{(a)} + \underbrace{2F_t^\top P_{t|t-1}^{-1}P_{t|t}\left(\nabla_{\hat{x}_{t|t}}\log\mathcal{N}\right)\left(z_t - H_t\hat{x}_{t|t-1}\right)^\top S_t^{-1}H_tF_t}_{(b)}\\
&\underbrace{+F_t^\top H_t^\top S_t^{-1}H_tF_t - F_t^\top H_t^\top S_t^{-1}\left(z_t - H_t\hat{x}_{t|t-1}\right)\left(z_t - H\hat{x}_{t|t-1}\right)^\top S_t^{-1}H_tF_t}_{(c)}
\end{aligned}$$

$$\begin{aligned}
=&(a)P_{t-1|t-1}^{-1}\underbrace{\underbrace{P_{t-1|t-1}F_t^\top P_{t|t-1}^{-1}}_{G_{t-1}}\left(P_{t|T} - P_{t|t-1}\right)\underbrace{P_{t|t-1}^{-1}F_tP_{t-1|t-1}}_{G_{t-1}^\top}}_{P_{t-1|T} - P_{t-1|t-1}}\cancel{P_{t-1|t-1}^{-1} - F_t^\top P_{t|t-1}^{-1}P_{t|t-1}H_t^\top S_t^{-1}H_tP_{t|t-1}^{-1}P_{t|t-1}F_t}\\
&+P_{t-1|t-1}^{-1}\underbrace{\underbrace{P_{t-1|t-1}F_t^\top P_{t|t-1}^{-1}}_{G_{t-1}}\left(\hat{x}_{t|T} - \hat{x}_{t|t-1}\right)}_{\left(\hat{x}_{t-1|T} - \hat{x}_{t-1|t-1}\right)}\left(\hat{x}_{t|T} - \hat{x}_{t|t-1}\right)^\top\underbrace{P_{t|t-1}^{-1}F_tP_{t-1|t-1}}_{G_{t-1}^\top}P_{t-1|t-1}^{-1}\\
&\cancel{+F_t^\top P_{t|t-1}^{-1}K_t\left(z_t - H_t\hat{x}_{t|t-1}\right)\left(z_t - H_t\hat{x}_{t|t-1}\right)^\top K_t^\top P_{t|t-1}^{-1}F_t - 2F_t^\top P_{t|t-1}^{-1}K_t\left(z_t - H_t\hat{x}_{t|t-1}\right)\left(\hat{x}_{t|T} - \hat{x}_{t|t-1}\right)^\top P_{t|t-1}^{-1}F_t}\\
&(b) + \cancel{2F_t^\top P_{t|t-1}^{-1}P_{t|t}P_{t|t}^{-1}\left(\hat{x}_{t|T} - \hat{x}_{t|t}\right)\left(z_t - H_t\hat{x}_{t|t-1}\right)^\top K_t^\top P_{t|t-1}^{-1}F_t}\\
&\cancel{(c) + F_t^\top H_t^\top S_t^{-1}H_tF_t - F_t^\top H_t^\top S_t^{-1}\left(z_t - H_t\hat{x}_{t|t-1}\right)\left(z_t - H\hat{x}_{t|t-1}\right)^\top S_t^{-1}H_tF_t}\\
=&-2\left[.5P_{t-1|t-1}^{-1} - .5P_{t-1|t-1}^{-1}\left(P_{t-1|T} + \left(\hat{x}_{t-1|T} - \hat{x}_{t-1|t-1}\right)\left(\hat{x}_{t-1|T} - \hat{x}_{t-1|t-1}\right)^\top\right)P_{t-1|t-1}^{-1}\right]
\end{aligned}$$

$$
\begin{aligned}
&\text{where } (a) = F_t^\top P_{t|t-1}^{-1} P_{t|t} \left(\nabla_{P_{t|t}} \log \mathcal{N}\right) P_{t|t} P_{t|t-1}^{-1} F_t \\
=&F_t^\top P_{t|t-1}^{-1} \cancel{P_{t|t}} \cancel{P_{t|t}^{-1}} \left(P_{t|T} - P_{t|t} + \left(\hat{x}_{t|T} - \hat{x}_{t|t}\right)\left(\hat{x}_{t|T} - \hat{x}_{t|t}\right)^\top\right) \cancel{P_{t|t}^{-1}} \cancel{P_{t|t}} P_{t|t-1}^{-1} F_t \\
=F_t^\top P_{t|t-1}^{-1} \Bigg(& P_{t|T} - \underbrace{P_{t|t-1} - P_{t|t-1} H_t^\top S_t^{-1} H_t P_{t|t-1}}_{P_{t|t}} + \Bigg(\hat{x}_{t|T} \underbrace{- \hat{x}_{t|t-1} - K_t\left(z_t - H_t \hat{x}_{t|t-1}\right)}_{-\hat{x}_{t|t}} \Bigg) \left(\hat{x}_{t|T} - \hat{x}_{t|t-1} - K_t\left(z_t - H_t \hat{x}_{t|t-1}\right)\right)^\top \Bigg) P_{t|t-1}^{-1} F_t
\end{aligned}
$$

C.2 BACKPROPAGATING HMM SMOOTHING DENSITY THROUGH TIME

In the following, we will develop the backpropagation algorithm for calculating the gradient of loss with respect to the forward/filter probabilities in a hidden Markov model. Our objective is to leverage this derivation to provide additional insights into the equivalence between the belief propagation algorithm (Bishop, 2006) and the gradient backpropagation algorithm.

We formulate the filter probabilities of a hidden Markov model as $\log \alpha(\mathbf{x}_t; \boldsymbol{\alpha}_t) = \mathbf{x}_t^\top \log \boldsymbol{\alpha}_t - \exp(\mathbf{1}^\top \log \boldsymbol{\alpha}_t)$, where the (canonical) parameters are $\log \boldsymbol{\alpha}_t \overset{\text{def.}}{=} (\log p(x_t = i | y_{1,\ldots,t}))_i$, $\mathbf{x}_t$ is the 1-hot encoding of the categorical distribution, and $\mathbf{1}$ is a column vector with all elements being 1. The gradient of the filter distribution over mean parameters is $\nabla_{\log \boldsymbol{\alpha}_t} \log \alpha(\mathbf{x}_t; \boldsymbol{\alpha}_t) = \mathbf{x}_t - \boldsymbol{\alpha}_t$. The differential form is $d_{\log \boldsymbol{\alpha}_t} \log \alpha(\mathbf{x}_t; \boldsymbol{\alpha}_t) = \mathbf{x}_t^\top d \log \boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\top d \log \boldsymbol{\alpha}_t$. So by theorem 1, the gradient of the log-likelihood with respect to the canonical parameters is $\nabla_{\log \boldsymbol{\alpha}_t} \log p(y_{1,\ldots,T}) = \mathbf{E}_{\gamma(x_t)} \nabla_{\log \boldsymbol{\alpha}_t} \log \alpha(x_t; \boldsymbol{\alpha}_t) = \boldsymbol{\gamma}_t - \boldsymbol{\alpha}_t$, where $\boldsymbol{\gamma}_t \overset{\text{def.}}{=} (p(x_t = i | y_{1,\ldots,T}))_i$.

Theorem 4. *The gradient of loss over filter parameters in a hidden Markov model can be recursively computed in the following way.*

$$
\nabla_{\log \alpha_{t-1,x_{t-1}}} \log p(y_{1,\cdots,T}) = \sum_{x_t} p(x_{t-1} | x_t, y_{1;t-1}) \nabla_{\log \alpha_{t,x_t}} \log p\left(y_{1:T}\right) + p(x_{t-1} | y_{1,\ldots,t}) - \alpha(x_{t-1}) \tag{15}
$$

Proof. The perturbation of the log likelihood with respect to the filter density in differential form is

$$
d_{\log \boldsymbol{\alpha}_t} \log p(y_{1,\ldots,T}) = \mathbf{E}_{\gamma(x_t)} d \log \alpha(x_t; \log \boldsymbol{\alpha}_t) = \left(\nabla_{\log \boldsymbol{\alpha}_t} \log p(y_{1,\ldots,T})\right)^\top \left(d \log \boldsymbol{\alpha}_t\right).
$$

We use the relationship in the forward propagation to transform it into $d_{\log \boldsymbol{\alpha}_{t-1}} \log p(y_{1,\ldots,T}) = \left(\nabla_{\log \boldsymbol{\alpha}_{t-1}} \log p(y_{1,\ldots,T})\right)^\top \left(d \log \boldsymbol{\alpha}_{t-1}\right)$ and subsequently find the backpropagation form from reading off the coefficient of $d \log \boldsymbol{\alpha}_{t-1}$.

$$
\begin{aligned}
& d \log p\left(y_{1:T}\right) = \left(\gamma_t - \alpha_t\right)^\top \cdot d \log \alpha_t \\
=& \sum_{x_t} \left(\gamma(x_t) - \alpha(x_t)\right) d \log \alpha\left(x_t\right), \text{ where } \nabla_{\log \alpha_{t,x_t}} = \left(\gamma(x_t) - \alpha(x_t)\right) \\
=& \sum_{x_t} \nabla_{\log \alpha_{t,x_t}} \log p(y_{1,\cdots,T}) \sum_{x_{t-1}} d \log \alpha(x_{t-1}) \left(p(x_{t-1} | x_t, y_{1;t-1}) - p(x_{t-1} | y_{1:t})\right) \\
=& \sum_{x_{t-1}} d \log \alpha(x_{t-1}) \left[\sum_{x_t} p(x_{t-1} | x_t, y_{1;t-1}) \nabla_{\log \alpha_{t,x_t}} - p(x_{t-1} | y_{1:t}) \underbrace{\sum\nolimits_{x_t} \nabla_{\log \alpha_{t,x_t}}}_{=0}\right] \\
& d \log_p(y_t | y_{1,\ldots,t-1}) = \sum_{t-1} d \log \alpha(x_{t-1}) \left[p(x_{t-1} | y_{1,\ldots,t}) - \alpha(x_{t-1})\right] \\
\Rightarrow & \nabla_{\log \alpha_{t-1,x_{t-1}}} \log p(y_{1,\cdots,T}) = \nabla_{\log \alpha_{t-1,x_{t-1}}} \log p\left(y_{1:T}\right) + \nabla_{\log \alpha_{t-1,x_{t-1}}} \log_p(y_t | y_{1,\ldots,t-1}) \\
=& \sum_{x_t} p(x_{t-1} | x_t, y_{1;t-1}) \nabla_{\log \alpha_{t,x_t}} \log p(y_{1,\cdots,T}) + p(x_{t-1} | y_{1,\ldots,t}) - \alpha(x_{t-1})
\end{aligned}
$$

□

As a sanity check, we demonstrate that substituting $\nabla_{\log \boldsymbol{\alpha}_t} \log p(y_{1,\ldots,T}) = \boldsymbol{\gamma}_t - \boldsymbol{\alpha}_t$ into the recursive relationship in Eq. 15 yields $\nabla_{\log \boldsymbol{\alpha}_{t-1}} \log p(y_{1,\ldots,T}) = \boldsymbol{\gamma}_{t-1} - \boldsymbol{\alpha}_{t-1}$. This confirms that the gradient of loss with respect to filter probability parameters computed through backpropagation aligns with its analytical form.

$$
\begin{aligned}
&\nabla_{\log \alpha_{t-1,x_{t-1}}} \log p(y_{1,\cdots,T}) \\
=&\sum_{x_t} \nabla_{\log \alpha_{t,x_t}} \cdot p(x_{t-1}|x_t, y_{1;t-1}) + p(x_{t-1}|y_{1,\ldots,t}) - \alpha(x_{t-1}) \\
=&\sum_{x_t} (\gamma(x_t) \cancel{- \alpha(x_t)}) \cdot p(x_{t-1}|x_t, y_{1;t-1}) \cancel{+ p(x_{t-1}|y_{1,\ldots,t})} - \alpha(x_{t-1}) \\
=&\gamma(x_{t-1}) - \alpha(x_{t-1}).
\end{aligned}
$$

D NATURAL GRADIENT IN A GAUSSIAN LINEAR BAYESIAN NN

The natural gradient, which reflects the difference between mean/canonical parameters in posterior and prior distributions, is implicitly defined by Theorem 1. In the subsequent sections, we explicitly derive this natural gradient concerning the cross-entropy loss and synaptic weights when approximating a Bayesian neural network as a Gaussian linear graphical model. Our approach involves approximating loss perturbations with respect to perturbations in the mean and covariance of synaptic weights to the second order. By setting the derivative of the loss perturbation with respect to these perturbations to zero and confirming the positive definiteness of the Hessian matrix, we obtain the natural gradient, which optimally accounts for local curvature. Finally, we establish a connection between natural gradients and gradients in the context of a Gaussian linear Bayesian neural network.

By Theorem 1, the perturbation of loss with respect to the perturbations of weight mean and covariance is $-d\log p(\mathbf{y}|\mathbf{x}) = -\mathbf{E}_{\gamma(\mathbf{w}|\mathbf{x},\mathbf{y})} d\log \mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w})$, where $p(\mathbf{y}|\mathbf{x}) = \mathbf{E}_{\mathbf{w}\sim\mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w})} p(\mathbf{y}|\mathbf{x},\mathbf{w})$ in a Gaussian linear Bayesian neural network, and $\mathbf{w}$ is the weights. We approximate $d\log\mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w})$ to the second order.

$$
\begin{aligned}
& d\log\mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w}) \\
\approx & -.5\,\mathrm{Tr}\left(\boldsymbol{\Sigma}_\mathbf{w}^{-1} d\boldsymbol{\Sigma}_\mathbf{w}\right) + 5(\mathbf{w}-\boldsymbol{\mu}_\mathbf{w})\boldsymbol{\Sigma}_\mathbf{w}^{\top} d\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\Sigma}_\mathbf{w}^{-1}(\mathbf{w}-\boldsymbol{\mu}_\mathbf{w}) + d\boldsymbol{\mu}_\mathbf{w}^{\top}\boldsymbol{\Sigma}_\mathbf{w}^{-1}(\mathbf{w}-\boldsymbol{\mu}_\mathbf{w}) \\
& -.5\,\mathrm{Tr}\left(\boldsymbol{\Sigma}_\mathbf{w}^{-1} d\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\Sigma}_\mathbf{w}^{-1} d\boldsymbol{\Sigma}_\mathbf{w}\right) - d\boldsymbol{\mu}_\mathbf{w}^{\top}\boldsymbol{\Sigma}_\mathbf{w}^{-1} d\boldsymbol{\mu}_\mathbf{w}
\end{aligned}
$$

Setting the gradient of $\mathbf{E}_{\gamma(\mathbf{w}|\mathbf{x},\mathbf{y})} d\log\mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w})$ with respect to $d\boldsymbol{\mu}_\mathbf{w}$ and $d\boldsymbol{\Sigma}_\mathbf{w}$ to 0, we get the following natural gradients.

$$
\begin{aligned}
& \mathbf{E}_\gamma \nabla_{d\boldsymbol{\mu}_\mathbf{w}} d\log\mathcal{N} = \boldsymbol{\Sigma}_\mathbf{w}^{-1}(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w}) - \boldsymbol{\Sigma}_\mathbf{w}^{-1} d\boldsymbol{\mu}_\mathbf{w} \overset{\text{set}}{=} 0 \\
\Rightarrow & d\boldsymbol{\mu}_\mathbf{w} = (\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w}) \overset{\text{def.}}{=} \tilde{\nabla}_{\boldsymbol{\mu}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}) \\
& \mathbf{E}_\gamma \nabla_{d\boldsymbol{\Sigma}_\mathbf{w}} d\log\mathcal{N} = -5\boldsymbol{\Sigma}_\mathbf{w}^{-1} + .5\boldsymbol{\Sigma}_\mathbf{w}^{-1}(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w})(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w})^{\top}\boldsymbol{\Sigma}_\mathbf{w}^{-1} - .5\boldsymbol{\Sigma}_\mathbf{w}^{-1} d\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\Sigma}_\mathbf{w}^{-1} \overset{set}{=} 0 \\
\Rightarrow & \mathbf{E}_\gamma d\boldsymbol{\Sigma}_\mathbf{w} = (\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w})(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w})^{\top} - \boldsymbol{\Sigma}_\mathbf{w} \overset{\text{def.}}{=} \tilde{\nabla}_{\boldsymbol{\Sigma}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x})
\end{aligned}
$$

The Hessian of $\mathbf{E}_{\gamma(\mathbf{w}|\mathbf{x},\mathbf{y})} d\log\mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w})$ with respect to $d\boldsymbol{\mu}_\mathbf{w}$ and $d\boldsymbol{\Sigma}_\mathbf{w}$ is positive definite for non-degenerative weights covariance. So the natural gradient is optimal with the local curvature taken into consideration.

$$
\mathbf{E}_\gamma \nabla_{d\boldsymbol{\mu}_\mathbf{w}\otimes d\boldsymbol{\mu}_\mathbf{w}} d\log\mathcal{N} = -\boldsymbol{\Sigma}_\mathbf{w}^{-1},\ \mathbf{E}_\gamma \nabla_{d\boldsymbol{\Sigma}_\mathbf{w}\otimes d\boldsymbol{\Sigma}_\mathbf{w}} d\log\mathcal{N} = -\boldsymbol{\Sigma}_\mathbf{w}^{-1}\otimes\boldsymbol{\Sigma}_\mathbf{w}^{-1},\ \mathbf{E}_\gamma \nabla_{d\boldsymbol{\mu}_\mathbf{w}\otimes d\boldsymbol{\Sigma}_\mathbf{w}} d\log\mathcal{N} = 0.
$$

Since the gradient of loss with respect to weight mean and variance is

$$
\begin{aligned}
& -\nabla_{\boldsymbol{\mu}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}) = -\boldsymbol{\Sigma}_\mathbf{w}^{-1}(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w}), \\
& -\nabla_{\boldsymbol{\Sigma}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}) = -\boldsymbol{\Sigma}_\mathbf{w}^{-1}\left[(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w})(\mathbf{E}_\gamma\mathbf{w} - \boldsymbol{\mu}_\mathbf{w})^{\top} - \boldsymbol{\Sigma}_\mathbf{w}\right]\boldsymbol{\Sigma}_\mathbf{w}^{-1},
\end{aligned}
$$

(which can be read off from the coefficients of $d\boldsymbol{\mu}_\mathbf{w}^{\top}$ and $d\boldsymbol{\Sigma}_\mathbf{w}$ in the second order approximation of $d\log\mathcal{N}(\mathbf{w};\boldsymbol{\mu}_\mathbf{w},\boldsymbol{\Sigma}_\mathbf{w})$ above,) natural gradient and gradient have the following relationship:

$$
\tilde{\nabla}_{\boldsymbol{\mu}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}) = \boldsymbol{\Sigma}_\mathbf{w}\cdot\nabla_{\boldsymbol{\mu}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}), \quad \tilde{\nabla}_{\boldsymbol{\Sigma}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}) = \boldsymbol{\Sigma}_\mathbf{w}\cdot\nabla_{\boldsymbol{\Sigma}_\mathbf{w}} \log p(\mathbf{y}|\mathbf{x})\cdot\boldsymbol{\Sigma}_\mathbf{w}.
$$

Therefore, the natural gradient with respect to weight mean and variance in the cross-entropy loss captures the difference between these parameters in the posterior and prior distributions. Updating weight distribution parameters using a fixed learning rate of one mirrors the fixed-point algorithm seen in graphical models. Additionally, updating these parameters through stochastic gradient descent with momentum and natural gradient can be seen as a method for approximating the posterior weight distribution by aggregating information from multiple mini-batches.

E EXPERIMENT DETAILS

E.1 NEURAL NETWORK ARCHITECTURES

We implement neural network architectures based on the original papers with reference to open-access implementations. Below, we provide quotations from these papers along with our comments.

E.1.1 DENSENET-BC [L=100, K=12](HUANG ET AL., 2017)

On all datasets except ImageNet, the DenseNet used in our experiments has three dense blocks that each has an equal number of layers. Before entering the first dense block, a convolution with 16 (or twice the growth rate for DenseNet-BC [note: convolution with $2 \times 16 = 32$ features for DenseNet-Bottleneck+Compression]) output channels is performed on the input images. For convolutional layers with kernel size 3×3, each side of the inputs is zero-padded by one pixel to keep the feature-map size fixed. We use 1×1 convolution followed by 2×2 average pooling as transition layers between two contiguous dense blocks. At the end of the last dense block, a global average pooling is performed and then a softmax classifier is attached. The feature-map sizes in the three dense blocks are $32 \times 32, 16 \times 16$, and 8×8, respectively. We experiment with the basic DenseNet structure with configurations $\{L = 40, k = 12\}, \{L = 100, k = 12\}$ [note: growth rate $= 12$. among the 100 convolutional layers, 4 are between input, 3 dense blocks, and outputs, and the rest $100 - 4 = 96$ blocks are distributed in 3 dense blocks. since each dense block is composed of a number of convolution blocks with one 1×1 convolution layer and one 3×3 convolution layer, each dense block has $96/3/2 = 16$ convolution blocks] and $\{L = 100, k = 24\}$. For DenseNet- BC, the networks with configurations $\{L = 100, k = 12\}, \{L = 250, k = 24\}$ and $\{L = 190, k = 40\}$ are evaluated.

In our experiments on ImageNet, we use a DenseNet-BC structure with 4 dense blocks on 224×224 input images. The initial convolution layer comprises 2k convolutions of size 7×7 with stride 2 ; the number of feature-maps in all other layers also follow from setting k. The exact network configurations we used on ImageNet are shown in Table 1.

All the networks are trained using stochastic gradient descent (SGD). On CIFAR and SVHN we train using batch size 64 for 300 and 40 epochs, respectively. The initial learning rate is set to 0.1 , and is divided by 10 at 50% and 75% of the total number of training epochs. On ImageNet, we train models for 90 epochs with a batch size of 256. The learning rate is set to 0.1 initially, and is lowered by 10 times at epoch 30 and 60.

Following [8], we use a weight decay of 10^{-4} and a Nesterov momentum [35] of 0.9 without dampening. We adopt the weight initialization introduced by [10]. ... we add a dropout layer [33] after each convolu- tional layer (except the first one) and set the dropout rate to 0.2. The test errors were only evaluated once for each task and model setting.

Densenet architecture for ImageNet can also be confirmed from tf . keras . applications. DenseNet 121 . The code is here: https://github.com/keras-team/keras/blob/v2.12.0/keras/applications/densenet.py#L63

E.1.2 RESNET (HE ET AL., 2016)

The network inputs are 32×32 images, with the per-pixel mean subtracted. The first layer is 3×3 convolutions. Then we use a stack of $6n$ layers with 3×3 convolutions on the feature maps of sizes $\{32, 16, 8\}$ respectively, with $2n$ layers for each feature map size. The numbers of filters are $\{16, 32, 64\}$ respectively. The subsampling is performed by convolutions with a stride of 2 . The network ends with a global average pooling, a 10-way fully-connected layer, and softmax. There are totally $6n + 2$ stacked weighted layers. The following table summarizes the architecture:

output map size	32 $\times$32	16 $\times$16	8×8
# layers	1 $+2n$	$2n$	$2n$
# filters	16	32	64

When shortcut connections are used, they are connected to the pairs of 3×3 layers (totally $3n$ shortcuts). On this dataset we use identity shortcuts in all cases (i.e., option A), so our residual models have exactly the same depth, width, and number of parameters as the plain counterparts.

We use a weight decay of 0.0001 and momentum of 0.9 , and adopt the weight initialization in [13] and BN [16] but with no dropout. These models are trained with a minibatch size of 128 on two GPUs. We start with a learning rate of 0.1 , divide it by 10 at 32k and 48k iterations, and terminate training at 64k iterations, which is determined on a $45\text{k}/5\text{k}$ train/val split. We follow the simple data augmentation in [24] for training: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. For testing, we only evaluate the single view of the original 32×32 image.

Our implementation for ImageNet follows the practice in [21, 41]. The image is resized with its shorter side randomly sampled in [256, 480] for scale augmentation [41]. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [21]. The standard color augmentation in [21] is used. We adopt batch normalization (BN) [16] right after each convolution and before activation, following [16]. We initialize the weights as in [13] and train all plain/residual nets from scratch. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. We use a weight decay of 0.0001 and a momentum of 0.9 . We do not use dropout [14], following the practice in [16]. In testing, for comparison studies we adopt the standard 10-crop testing [21]. For best results, we adopt the fullyconvolutional form as in [41, 13], and average the scores at multiple scales (images are resized such that the shorter side is in $\{224, 256, 384, 480, 640\}$). .... See Table 1 for detailed architectures.

E.1.3 EFFICIENTNET B0 (TAN & LE, 2019)

The implementation follows keras reference implementation. We scaled input to $(224, 224)$.

```
inputs = layer_input(shape = dim(cifar10$train$x)[-1])
efficientNetB0.cifar10 = tf$keras$applications$EfficientNetB0(
  input_tensor = inputs %>%
  layer_lambda(function(img) tf$image$resize(img, c(224L, 224L))),
  include_top=TRUE,
  weights=NULL,
  classes=10L
)
efficientNetB0.cifar10$compile(
  loss='categorical_crossentropy',
  optimizer= 'adam',
  metrics=list('accuracy')
)
mc = callback_model_checkpoint(
  'efficientNetB0_CIFAR10.ckpt',
  monitor = 'val_accuracy',
  save_best_only = TRUE,
  mode = 'auto',
  save_weights_only = TRUE
)

tf$keras$utils$plot_model(
  efficientNetB0.cifar10,
  to_file='efficientNetB0_CIFAR10.png',
  show_shapes=TRUE,
  show_layer_names=TRUE,
  show_layer_activations=TRUE
)
IRdisplay::display_png(file = 'efficientNetB0_CIFAR10.png')
```

E.1.4 WIDERESNET-28-10 (ZAGORUYKO & KOMODAKIS, 2016)

Compared to the original architecture [11] in [13] the order of batch normalization, activation and convolution in residual block was changed from conv-BN-ReLU to BN-ReLU-conv. As the latter was shown to train faster and achieve better results we don't consider the original version.

The general structure of our residual networks is illustrated in table 1: it consists of an initial convolutional layer conv1 that is followed by 3 groups (each of size N) of residual blocks conv2, conv3 and conv4, followed by average pooling and final classification layer. The size of conv1 is fixed in all of our experiments, while the introduced widening factor k scales the width of the residual blocks in the three groups conv24 (e.g., the original ≪basic≫ architecture is equivalent to $k = 1$).

group name	output size	block type $= B(3, 3)$
conv1	32×32	$[3 \times 3$ $16]$
conv2	32×32	$\begin{bmatrix} 3 \times 3, \\ 16 \times \mathrm{k} \\ 3 \times 3, \\ 16 \times \mathrm{k} \end{bmatrix} \times \mathrm{N}$
conv3	16×16	$\begin{bmatrix} 3 \times 3, \\ 32 \times \mathrm{k} \\ 3 \times 3, \\ 32 \times \mathrm{k} \end{bmatrix} \times \mathrm{N}$
conv4	8×8	$\begin{bmatrix} 3 \times 3, \\ 64 \times \mathrm{k} \\ 3 \times 3, \\ 64 \times \mathrm{k} \end{bmatrix} \times \mathrm{N}$
avg-pool	1×1	$[8 \times 8]$

As widening increases the number of parameters we would like to study ways of regularization. Residual networks already have batch normalization that provides a regularization effect, however it requires heavy data augmentation, which we would like to avoid, and it's not always possible. We add a dropout layer into each residual block between convolutions as shown in fig. 1(d) and after ReLU to perturb batch normalization in the next residual block and prevent it from overfitting. In very deep residual networks that should help deal with diminishing feature reuse problem enforcing learning in different residual blocks. ... We trained networks with dropout inserted into residual block between convolutions on all datasets. We used cross-validation to determine dropout probability values, 0.3 on CIFAR and 0.4 on SVHN. Also, we didn't have to increase number of training epochs compared to baseline networks without dropout.

In all our experiments we use SGD with Nesterov momentum and cross-entropy loss. The initial learning rate is set to 0.1 , weight decay to 0.0005 , dampening to 0 , momentum to 0.9 and minibatch size to 128 . On CIFAR learning rate dropped by 0.2 at 60,120 and 160 epochs and we train for total 200 epochs. On SVHN initial learning rate is set to 0.01 and we drop it at 80 and 120 epochs by 0.1 , training for total 160 epochs. Our implementation is based on Torch [6]. We use [21] to reduce memory footprints of all our networks. For ImageNet experiments we used fb. resnet. torch implementation [10]. Our code and models are available at https://github.com/szagoruyko/wide-residual-networks. some implementations at github by the authors and others

- https://github.com/szagoruyko/wide-residual-networks
- https://github.com/titu1994/Wide-Residual-Networks/
- https://github.com/paradoxysm/wideresnet

E.1.5 MLP-MIXER (TOLSTIKHIN ET AL., 2021)

Tolstikhin, Ilya O., Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung et al. "Mlp-mixer: An all-mlp architecture for vision." Advances in neural information processing systems 34 (2021): 24261-24272.

E.2 MORE DETAILS ON VISUALIZING LEARNED JOINT OBSERVATION-LABEL DISTRIBUTION OF BOSTON HOUSING DATA

We present a visualization of the joint observation-label distribution generated by a Bayesian neural network through Monte Carlo simulations and 2D projections (Fig. 2). Our approach begins with minor perturbations to feature-label pairs in the training dataset, followed by evolving a Hamiltonian Monte Carlo chain with adaptive step-size. This chain targets the uncalibrated probability and uses the Gelman-Rubin diagnostic to ensure convergence. The outcome is a simulated collection of feature-label patterns that the model anticipates, offering insights into typical label observations and potential losses.

In Fig. 2 (a) and (c), we project this distribution to compare predicted versus actual labels. Ideally, a model should concentrate its probability mass along the diagonal, indicating the likelihood of an observation, particularly when predictions deviate from training. In Fig. 2 (a), a Bayesian neural network trained with our algorithm exhibits a tight observation-label correlation along the diagonal. Conversely, Fig. 2 (c) shows diverging predictions from two independently trained neural networks, especially in data-sparse regions, reflecting their unique generalizations.

In Fig. 2 (b) and (d), we visualize observations along the two principal components accounting for the most variance. Here, blue dotted lines mark the 1-σ, 2-σ, and 3-σ confidence regions, with solid black lines for predictions. The Bayesian neural network's expected observations cluster in areas correlating with lower loss (Fig. 2 (b)), while non-Bayesian networks exhibit a more dispersed pattern.

To validate our approach, we simulate belief propagation in a probabilistic graphical model using an ensemble of non-Bayesian neural networks in a shared computational graph. We compare the filter and smoothing densities from standard neural networks against those in the Bayesian neural network, and likewise for sensitivities. This comparison serves as a sanity check, demonstrating the effectiveness of our method.

E.3 VISUALIZING CNN LEARNED DISTRIBUTIONS

Figure 3: The probability distribution of observations $\mathbf{x}$ as a Bayesian neural network with 3 densely connected layers learns to classify the mixture component assignment of an observation from a mixture of Gaussian distribution. The BNN is trained with Algorithm 1, and the input is sampled from Hamiltonian Monte Carlo to estimate the probability distribution of $\mathbf{x}$ induced by the BNN.

Figure 4: Images generated from a trained Bayesian DenseNet with Hamiltonian Monte-Carlo.