Variation Due to Regularization Tractably Recovers Bayesian Deep Learning

James McInerney
Netflix Research
New York, NY, USA
jmcinerney@netflix.com

Nathan Kallus
Netflix Research & Cornell University
New York, NY, USA
nkallus@netflix.com

Abstract

Uncertainty quantification in deep learning is crucial for safe and reliable decision-making in downstream tasks. Existing methods quantify uncertainty at the last layer or other approximations of the network which may miss some sources of uncertainty in the model. To address this gap, we propose an uncertainty quantification method for large networks based on *variation due to regularization*. Essentially, predictions that are more (less) sensitive to the regularization of network parameters are less (more, respectively) certain. This principle can be implemented by deterministically tweaking the training loss during the fine-tuning phase and reflects confidence in the output as a function of all layers of the network. We show that regularization variation (RegVar) provides rigorous uncertainty estimates that, in the infinitesimal limit, exactly recover the Laplace approximation in Bayesian deep learning. We demonstrate its success in several deep learning architectures, showing it can scale tractably with the network size while maintaining or improving uncertainty quantification quality. Our experiments across multiple datasets show that RegVar not only identifies uncertain predictions effectively but also provides insights into the stability of learned representations.

1 INTRODUCTION

Bayesian neural networks, where a prior is placed on model weights, offer an opportunity to understand uncertainty and improve model robustness (Wilson, 2020; Papamarkou et al., 2024). However, exact computation of the model-weight posterior and predictive distributions is prohibitive given size and complexity. It is instead appealing to approximate it using the maximum *a posteriori* (MAP) solution and the curvature thereat. However, even this curvature, characterized by a Hessian, can be challenging to compute.

The Laplace approximation of the posterior distribution uses the (negative) Hessian of the log joint distribution at the MAP (Mackay, 1992). For high dimensional θ – common in deep neural networks – these steps are a prohibitive computational bottleneck even for an approximate posterior. One remedy is further approximation of the Hessian, but this may ignore important directions of curvature.

In this paper we propose using the principle of variation due to regularization (RegVar), which directly approximates the uncertainty of the predicted mean in the Laplace approximation without explicitly computing and inverting the Hessian (while still assuming it exists). Instead, one additional point estimate is required, the *prediction-regularized* MAP, derived from the network objective with a small amount of regularization added. Under the assumptions that the Laplace approximation is used, we find that the rescaled difference in network outputs given by the MAP and the prediction-regularized MAP recovers the Laplace variance.

Our contributions in this paper are as follows:

- We develop a set of novel methods around regularization variation (RegVar) and formalize it as the predictive variance of linearized Laplace.
- In RegVar, variance is derived from the change in the MAP prediction when adjusting the regularizer. The potential appeal of RegVar is the simplicity of the method (see Figure 1) and implementation

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

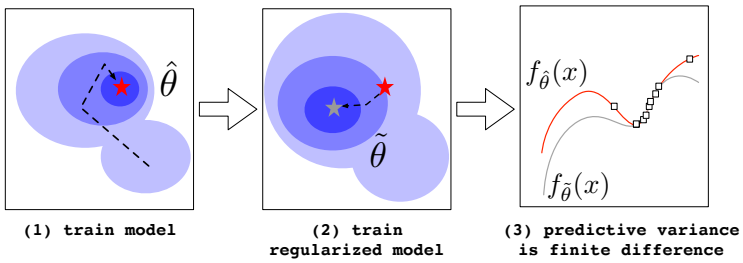


Figure 1: Summary of the regularization-based approach to uncertainty quantification that underlies RegVar. **Step 1:** fit a predictive model by MAP; **step 2:** fit another model using an infinitesimally regularized MAP; **step 3:** the difference between the predictions of these two models, normalized by the amount of regularization, is exactly the estimate of the predictive variance.

```
// training step to update model parameters theta with training sample (x, y)
// n: training data size, lam: scalar hyperparameter
f = model(x)
loss = cross_entropy(f, y) - log_prior(theta) / n + lam * f.abs().mean() / n
loss.backward()
optimizer.step()
...

//inference time for input x_test
f_pred_var = (model(x_test) - f_pred_mean).abs() / lam
```

Figure 2: Example of fine-tuning a model using regularization variation (RegVar). The code required to extend existing fine-tuning to implement RegVar is given in blue. Any auto-differentiation package is sufficient, here we use PyTorch (Paszke et al., 2017).

(see Figure 2 for an example).

- We show that RegVar scales up to large networks and compare it to several other popular Bayesian deep learning approaches.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we set up the key aspects of the Laplace approximation and connect it to regularization variation in Section 4. Empirical evaluation is explored in Section 5 before discussing conclusions and future work in Section 6.

2 RELATED WORK

The Laplace approximation was first formulated in BDL for small-scale neural networks (MacKay, 1992). Since then, larger architectures have necessitated Hessian approximations, including Gauss-Newton (Foresee and Hagan, 1997) and Kronecker factorization that places a block diagonal structure on the covariate in line with assumed independence between the layers of a network (Martens and Grosse, 2015).

In recent years, with the aforementioned scalable approximations and renewed appreciation of the advantages of Laplace as a *post-hoc* uncertainty method (Daxberger et al., 2021), a number of new directions were addressed, including relating neural network loss functions to Gaussian process inference (Khan et al., 2019) and exploring the advantages of locally linearized Laplace in out-of-distribution evaluation (Foong et al., 2019). Underfitting in Laplace (Lawrence, 2001) is mitigated in the GGN approximation by switching to the generalized linear model implied by the local linearization (Immer et al., 2021). Further scaling with linearized Laplace applies neural tangent kernels (Deng et al., 2022) and variational approximation (Ortega et al., 2023). For large models, Laplace can be applied to a low-rank adaptor Yang et al. (2024).

Beyond Laplace-based methods, a number of approaches have been proposed for scalable Bayesian inference in neural networks. Given the breadth of this topic, we give a non-exhaustive account (see Papamarkou et al., 2024 for further discussion). Variational inference approximates the posterior with a parameterized distribution, enabling efficient optimization via stochastic gradient descent (Graves, 2011; Blundell et al., 2015; Shen et al., 2024). Stochastic gradient

Langevin dynamics and related approaches that take samples from the optimization trajectory of a neural network use gradient noise to approximate Bayesian updates without explicit posterior evaluation (Welling and Teh, 2011; Mandt et al., 2016; Maddox et al., 2019). Dropout-based uncertainty estimation interprets dropout training as an approximate Bayesian inference method (Gal and Ghahramani, 2016). Ensembles of independently trained networks use multiple predictions to quantify the uncertainty of black-box models (Lakshminarayanan et al., 2017; Osband and Van Roy, 2015).

Owing to the ubiquity of the Hessian of the objective and its computational demands, approaches to avoiding its direct evaluation are developed in areas of research distinct of Laplace. In particular, Hessian-free optimization uses the conjugate gradient method to iteratively update a matrix-vector product as part of an inner loop in Newton's method second-order optimization (Martens and Sutskever, 2011; Pearlmutter, 1994). Finally, Kallus and McInerney (2022) develop a frequentist approach establishing that infinitesimal regularization of the total log likelihood yields a variance estimate that is asymptotically equivalent to the delta method. While avoiding full Fisher matrix computation, this approach requires evaluating a perturbed estimator per test point and output dimension, leading to computational costs that scale accordingly. It is also a frequentist perspective, in contrast to the Bayesian approach developed in this paper.

3 BACKGROUND

Consider the maximum *a posteriori* (MAP) solution of a Bayesian neural network f_θ with parameters θ given data $\{(x_i, y_i)\}_{i=1}^n$,

$$\hat{\theta} \in \arg_\theta \max \mathcal{L}_\theta \tag{1}$$

$$\text{where } \mathcal{L}_\theta = \sum_{i=1}^{n} \log p(y_i \mid f_\theta(x_i)) + \log p(\theta), \tag{2}$$

for some observation likelihood $p(y \mid \cdot)$ and prior $p(\theta)$. This is equivalent to optimizing a loss (e.g., squared loss for Gaussian likelihood or cross-entropy for categorical likelihood) regularized by $\log p(\theta)$ and means that computing the MAP of a Bayesian neural network is akin to fitting a vanilla neural network.

Our starting point is the learning of parameters θ of a deep neural network f_θ as per Eq. 1 and 2. While the representational power and accuracy of the network may be high, in many applications it is crucial to also quantify the uncertainty of predictions, such as in autonomous vehicles, healthcare, or recommendations (Kendall and Gal, 2017; Leibig et al., 2017).

Bayesian deep learning (BDL) provides a framework for analyzing uncertainty in deep learning (Papamarkou et al., 2024). The key components are the prior distribution $p(\theta)$ and likelihood function $p(y \mid \theta, x)$ in a family of models indexed by θ, and an i.i.d. dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ assumed to be collected from a model in this family. The posterior distribution $p(\theta \mid \mathcal{D})$ may be flexibly used as the density (or mass) for the expectation of any downstream prediction depending on θ, e.g. expectation or variance of predicted mean, as part of a larger simulation involving fitted deep models, or interpreting the variance of weights for different layers of the network. The main evaluations of interest in BDL are the mean $\mathbb{E}_{p(\theta \mid \mathcal{D})}[f_\theta(x)]$ and (co)variance $\mathrm{Var}_{p(\theta \mid \mathcal{D})}[f_\theta(x)]$ of the prediction for a query vector (or set of query vectors) x. These will be the focus of our study in this paper.

BDL is doubly intractable. First, the normalizing factor in the Bayesian posterior is intractable for all but the most trivial networks as it entails a sum over high-dimensional θ. Many techniques have been developed to approximate the posterior – the most common being MAP estimation, variational inference, and Markov chain Monte Carlo – and these vary in their scalability and applicability to deep neural nets (Blundell et al., 2015; Ritter et al., 2018). A key desideratum for approximate inference that is often overlooked is the extent to which it is compatible with existing deep learning frameworks and implementations, which may be supported by years of model selection, tuning, and vast training budgets. The MAP point estimate is the most practical in this sense, and is the basis of the Laplace approximation. Second, even with an approximate posterior to hand, the posterior predictive $p(y \mid \mathcal{D}, x)$ depends on f_θ, meaning that the integral cannot be further simplified. Monte Carlo estimation using samples from the posterior is used to address this intractability.

Laplace Approximation Let q be the Laplace approximation of the posterior, defined as the second-order Taylor expansion around $\hat{\theta}$,

$$\log q(\theta) = \log p(\hat{\theta} \mid \mathcal{D}) - \frac{1}{2}(\theta - \hat{\theta})^\top P (\theta - \hat{\theta}), \tag{3}$$

where $P := -\nabla\nabla_\theta \log p(\theta \mid \mathcal{D})$. The first derivative does not appear in Eq. 3 because $\hat{\theta}$ is defined as a local optimum of the posterior, where it exists.

Noticing that Eq. 3 takes the form of a quadratic in θ, we see that $q(\theta) = \mathcal{N}(\hat{\theta}, P^{-1})$. If precision matrix P can be calculated and inverted, then the first intractability of approximating the posterior is addressed. In addition, the normal form of q also provides a closed-form solution to the model evidence, supporting model and

hyperparameter selection (MacKay, 1992). We next discuss precision calculation in more detail.

Precision Matrix Distribution $q(\theta)$ requires the covariance matrix which entails calculating and inverting the precision matrix P decomposed as,

$$\begin{aligned} P &= -\nabla\nabla_\theta \log p(\theta \mid \mathcal{D}) \\ &= \underbrace{-\nabla\nabla_\theta \log p(\theta)}_{\text{Gaussian prior} \implies \text{scalar matrix}} \underbrace{- \nabla\nabla_\theta \log p(\mathcal{D} \mid \theta)}_{\text{Hessian } H}. \end{aligned} \tag{4}$$

The prior term in Eq. 4 usually takes a simple form, e.g. constant scalar for Gaussian prior, so our focus from this point will be on H, the Hessian of the *likelihood*.

The Hessian of the likelihood term in Eq. 4 may be further decomposed as,

$$\begin{aligned} H = &\underbrace{\nabla_f \log p(y \mid f_\theta(x))}_{\text{residual}} \nabla\nabla_\theta f_\theta(x) + \\ &\underbrace{\nabla\nabla_f \log p(y \mid f_\theta(x))}_{\text{observation precision}} \quad \nabla_\theta f_\theta(x) \nabla_\theta f_\theta(x)^\top. \end{aligned} \tag{5}$$

In practice, the precision calculation of the Laplace approximation drops the first term – the network Hessian – in Eq. 5 due to computational constraints. This is known as generalized Gauss-Newton (GGN) and is proportional to the outer product of the network Jacobian (second term in Eq. 5). When GGN is used in approximate inference, it assumes the network takes the form of a generalized linear model. In Laplace, this is equivalent to a local linearization of the network prediction around $\hat{\theta}$,

$$\tilde{f}_\theta(x) = f_{\hat{\theta}}(x) + \nabla_\theta f_\theta(x)\Big|_{\hat{\theta}}^\top (\theta - \hat{\theta}), \tag{6}$$

which experimentally results in more accurate predictions than the original network $f_{\hat{\theta}}$ when used in the Monte Carlo average for evaluation (Foong et al., 2019; Immer et al., 2021). However, the Hessian, even after simplifying with GGN, is still unwieldy or prohibitive to compute, store, and invert for large networks with high dimensional θ.

4 REGULARIZATION VARIATION (REGVAR)

Against this background, we target the same mathematical object as the Laplace approximation, requiring the Hessian to exist but avoiding the need to calculate or invert it. To start, notice that an immediate consequence of Eq. 6 is,

$$\tilde{f}_\theta \sim \mathcal{N}(f_{\hat{\theta}}(x), \nabla_\theta f_{\hat{\theta}}(x)^\top P^{-1} \nabla_\theta f_{\hat{\theta}}(x)), \tag{7}$$

where $\nabla_\theta f_{\hat{\theta}} := \nabla_\theta f_\theta\Big|_{\hat{\theta}}$ is used for more compact notation. This is an instance of the Bayesian delta method (Wasserman, 2006), differing from the classic delta method by the inclusion of the prior term in P in Eq. 4.

The posterior predictive depends on the form of the observation likelihood function. For regression tasks, Gaussian observation noise with fixed standard deviation σ is the standard, resulting in posterior predictive equal to Eq. 7 plus additional variance term σ^2. For classification tasks, the posterior predictive is non-analytical, requiring Monte Carlo averaging over posterior samples. Notwithstanding the particular form of the posterior predictive distribution, the *epistemic uncertainty* of the network predictions – meaning, the error due to lack of knowledge – is represented by Eq. 7 and has considerable value in active learning, experimental design, and exploration-exploitation.

We set up the prediction-regularized joint distribution as equal to the joint distribution (Eq. 2) with an additional term proportional to the prediction at query point x,

$$\mathcal{L}_\theta{}^{(f(x),\lambda)} = \sum_{i=1}^{n} \log p(y_i \mid f_\theta(x_i)) + \log p(\theta) + \lambda f_\theta(x), \tag{8}$$

for some constant $\lambda \in \mathbb{R}$. The prediction-regularized MAP is defined as,

$$\hat{\theta}^{(f(x),\lambda)} \in \arg_\theta \max \mathcal{L}_\theta{}^{(f(x),\lambda)}. \tag{9}$$

Theorem 4.1. *The derivative of the prediction under the prediction-regularized MAP w.r.t. λ recovers the variance-covariance term in Eq. 7,*

$$\nabla_\lambda f_{\hat{\theta}^{(f(x),\lambda)}}(x)\Big|_{\lambda=0} = \nabla_\theta f_{\hat{\theta}}(x)^\top P^{-1} \nabla_\theta f_{\hat{\theta}}(x), \tag{10}$$

assuming $\hat{\theta}^{(f(x),\lambda)}$ and P^{-1} exist, and that $\mathcal{L}_\theta{}^{(f(x),\lambda)}$ is continuously differentiable w.r.t. θ.

Proof. See Appendix A. □

A parallel frequentist approach is known as the implicit delta method (Kallus and McInerney, 2022). The relationship in Eq. 10 can also be understood as an application of the implicit function theorem (IFT) by Cauchy, e.g. see (Lorraine et al., 2020) for a modern treatment of IFT.

A corollary of Theorem 4.1 is that detecting the local change in prediction as λ is increased (or decreased) from 0 yields the variance term of the linearized Laplace approximation in Eq. 7. The local change, i.e. the LHS

Algorithm 1 Regularization Variation (RegVar)

Input: network f_θ, fitted MAP $\hat{\theta}$, training data $\mathcal{D}$ of size n, evaluation inputs $\{\hat{x}_{i=1}^m\}$, scalar λ, step size schedule $\gamma(\cdot)$
Warm start $\hat{\theta}^{(\hat{r},\lambda)} \leftarrow \hat{\theta}$
Initialize step counter $j = 0$
repeat
  Sample training example $(x_i, y_i) \sim \mathcal{D}$
  Follow gradient $\hat{\theta}^{(\hat{r},\lambda)} \leftarrow \hat{\theta}^{(\hat{r},\lambda)} + n\gamma(j)\nabla_\theta \mathcal{L}_i$ // see Eq. 15
  $j \leftarrow j + 1$
until $\hat{\theta}^{(\hat{r},\lambda)}$ converges
Return: predictive variance function $\hat{\sigma}_f^2(x) := \frac{1}{\lambda}|f_{\hat{\theta}^{(\hat{r},\lambda)}}(x) - f_{\hat{\theta}}(x)|$

of Eq. 10, may be calculated by finite differences for some suitably small λ,

$$\frac{1}{\lambda}(f_{\hat{\theta}^{(f(x),\lambda)}}(x) - f_{\hat{\theta}}(x)) \underset{\lambda\to 0}{\to} \nabla_\lambda f_{\hat{\theta}^{(f(x),\lambda)}}(x)\Big|_{\lambda=0}. \quad (11)$$

It is natural to consider auto-differentiation to calculate the LHS of Eq. 10. From the fact that the optimum $\hat{\theta}^{(f(x),\lambda)}$ also changes with λ it can be seen that auto-differentiation must deal with complex operations on optima and is unlikely to lead to a computational advantage unless used in tandem with additional workarounds.

Putting it together, the regularization variation (RegVar) approximation is,

$$\tilde{f}_\theta \sim \mathcal{N}\left(f_{\hat{\theta}}(x), \frac{1}{\lambda}(f_{\hat{\theta}^{(f(x),\lambda)}}(x) - f_{\hat{\theta}}(x))\right), \quad (12)$$

and differs from linearized Laplace (Eq. 7) by the lack of an explicit Hessian. It is apparent by this comparison that RegVar trades the computational and storage expense of dealing with the Hessian for that of a pointwise approximation for each query x. This operation may be amortized into the network, which we discuss next.

4.1 Amortized Regularizer

RegVar targets the same predictive variance-covariance as the full linearized Laplace approximation. However, when evaluating the uncertainty of many predictions, the requirement to optimize to the f-objective specific to each test point is a bottleneck. In order to scale RegVar in the number of test points, we develop an amortized regularization approach that requires only a single model to represent the f-regularized parameters for a set of evaluation points.

Amortization is formulated as follows. Find the regularization term $\hat{r}$ for a finite set of evaluation inputs $\hat{\mathbf{x}} = \{\hat{x}_{1:m}\}$ that minimizes the average squared L2 norm error of the corresponding f-regularized parameters,

$$\hat{r} = \arg_r \min \sum_{i=1}^m \|\hat{\theta}^{(f(x_i),\lambda)} - \hat{\theta}^{(r,\lambda)}\|_2^2. \quad (13)$$

Eq. 13 facilitates the calculation of Eq. 12 using $\hat{\theta}^{(\hat{r},\lambda)}$ instead of $\hat{\theta}^{(f(x),\lambda)}$ for arbitrary x in the domain of f.

Theorem 4.2. *The optimal regularization term defined by Eq. 13 is $\hat{r} = \frac{1}{m}\sum_{i=1}^m f(\hat{x}_i)$ when $\hat{\theta}^{(f(x),\lambda)}$ and P^{-1} exist and $\mathcal{L}_\theta{}^{(f(x),\lambda)}$ is continuously differentiable w.r.t. θ.*

Proof. See Appendix A. □

Theorem 4.2 implies that embedding the average network output in the training objective minimizes the L2 norm of the f-regularized objective. Furthermore, we can absorb a sign term into λ that depends on the sign of the output function by defining,

$$\tilde{\lambda} = \lambda(-1)^{\mathbb{I}[f_\theta(x)<0]}, \quad (14)$$

where $\mathbb{I}[c]$ is the indicator function that evaluates to 1 when the condition c is true and to 0 otherwise. This equates to using the absolute values of both the RegVar regularizer and predictive variance. Putting it together leads to the amortized objective,

$$\mathcal{L}_i = \log p(y_i \mid f_\theta(x_i)) + \frac{1}{n}\log p(\theta) + \frac{\lambda}{nm}\|f_\theta(\hat{\mathbf{x}})\|_1, \quad (15)$$

where the final term in Eq. 15 is the (rescaled) sum of absolute values of the function applied to the m evaluation inputs. We emphasize that this sum of absolute values accommodates networks with multiple outputs, a crucial requirement for practical use.

A stochastic optimization algorithm for amortized RegVar is given in Algorithm 1. The procedure avoids the need to explicitly calculate the Hessian and instead optimizes one parameter vector $\hat{\theta}^{(\hat{r},\lambda)}$. The general form

of the objective optimized in the algorithm is Eq. 15. We next highlight a special case and a variant of Eq. 15, both of which admit simplified forms of RegVar.

Special Case: In-Sample Amortization When the train and test data distributions are the same, the data sampled to optimize the log joint probability of the model may also be used to regularize it, giving rise to the single-sample stochastic gradient,

$$\mathcal{L}_i^{(\text{IS})} = \log p(y_i \mid f_\theta(x_i)) + \frac{1}{n}\log p(\theta) + \frac{\lambda}{nm}|f_\theta(x_i)|. \tag{16}$$

Alternatively, a data augmentation approach is applicable to in-sample training when the likelihood is Gaussian (see Appendix B).

Variant: Parameter Uncertainty Quantification The same amortized regularization strategy also enables parameter uncertainty quantification by finding the regularizer $\hat{t}$ that minimizes the average squared L2 norm over the K dimensions of parameters,

$$\hat{t} = \arg_t \min \sum_{k=1}^{K} \|\hat{\theta}^{(\theta_k,\lambda)} - \hat{\theta}^{(t,\lambda)}\|_2^2. \tag{17}$$

Theorem 4.2 applies to Eq. 17 in combination with absolute value regularization to provide the amortized objective,

$$\mathcal{L}_i^{(\text{PU})} = \log p(y_i \mid f_\theta(x_i)) + \frac{1}{n}\log p(\theta) + \frac{\lambda}{nm}\|\theta\|_1, \tag{18}$$

which corresponds to a set of sparsity-inducing Laplace priors over the weights. Thus, when used in concert with a Laplace log prior $\log p(\theta)$, the regularization term for RegVar is absorbed, with λ perturbing the precision parameter. In this case, it is particularly convenient to implement RegVar for parameter uncertainty, as one needs only to adjust the strength of the existing regularizer, as explored in Appendix B.

5 EXPERIMENTS

We investigate the empirical performance of RegVar in quantifying uncertainty for large models. The goal is to validate the theoretical developments from Section 3 and compare to baselines in Bayesian deep learning. We next describe the experimental setup.

Experimental Setup The experiments use a Small GPT-3 Architecture language model with 125 million parameters and a Vision Transformer (Base) with 86 million parameters. To study how the methods scale, we also experiment with a larger XL GPT-3 Architecture comprising 1.3 billion parameters. For the language models, we load the pre-trained model and fine-tune it to **Wikitext** (wikitext-103-raw) comprising unprocessed text extracted from Wikipedia (Merity et al., 2016) and **IMDB** a binary sentiment classification dataset comprising movie reviews split evenly between positive and negative sentiments (Maas et al., 2011). For the vision model, we fine-tune the pre-trained model to **Places365** (places365), an image dataset of 365 scenes (Zhou et al., 2017). The fine-tuning language datasets were subsampled to 40k documents split into 80% train, 10% validation, and 10% test, while the vision dataset was subsampled to 4k images split into 50% train, 25% validation, and 25% test.

The language models were fine-tuned over 1 epoch with stochastic gradient descent using adaptive learning rates,[1] weight decay 0.01,[2] and batch size 4 with 4 gradient accumulation steps. These experiments were conducted on an instance configured with one NVIDIA A100 GPU with 40 GB memory, 12 CPU cores, and 143 GB of system memory. The vision model was fine-tuned over 5 epochs with the same settings without the gradient accumulation on an instance configured with 4 NVIDIA A10 GPUs each with 24 GB memory, 96 CPU cores, and 380 GB of system memory.

We compared a number of uncertainty quantification methods,

- **Maximum *a posteriori* (MAP)** optimizes the log joint distribution w.r.t. the network parameters (see Eq. 1).
- **Ensemble Last-Layer** performs MAP estimation with multiple heads at the last layer with a shared network before the last layer (Lakshminarayanan et al., 2017; Osband and Van Roy, 2015). During training, heads are randomly initialized and, for each training example, a head is sampled at random to receive an update. During inference, the mean and variance of the population of all head evaluations is used for uncertainty. Memory constraints prohibit approximations beyond the last layer and prevent applying ensembling to the XL GPT-3 architecture.
- **Monte Carlo Dropout (MC Dropout)** uses dropout to generate approximate samples of the network during inference time (Gal and Ghahramani, 2016). Since dropout is applied at every layer, it is the only other method (apart from

[1] Adam with learning rate 2×10^{-5} (Kingma and Ba, 2017).

[2] Equivalent to $\mathcal{N}(0, 50I)$ prior on network parameters.

Table 1: Held-Out Predictive Errors with GPT-3, XL GPT-3, and ViT Architectures

Method	GPT-3 (125M Param.)		XL GPT-3 (1.3B Param.)		ViT (86M Param.)
	Wikitext	IMDB	Wikitext	IMDB	Places365
Negative Log Likelihood (↓ is better) ±95% Confidence Interval					
Pre-Trained	9.781 ± 0.009	10.374 ± 0.012	8.228 ± 0.013	9.886 ± 0.015	5.900 ± 0.002
MAP	1.383 ± 0.027	9.681 ± 0.015	1.523 ± 0.030	9.206 ± 0.020	5.871 ± 0.012
Ensemble	3.925 ± 0.058	22.84 ± 0.005	—	—	5.885 ± 0.010
MC Dropout	1.410 ± 0.026	9.551 ± 0.013	1.495 ± 0.029	9.137 ± 0.019	5.878 ± 0.006
Laplace Diag	1.481 ± 0.021	9.090 ± 0.011	1.273 ± 0.021	8.797 ± 0.017	5.872 ± 0.026
Laplace Full	1.277 ± 0.021	8.774 ± 0.009	1.198 ± 0.021	8.047 ± 0.013	5.869 ± 0.001
RegVar (this paper)	1.143 ± 0.021	8.009 ± 0.008	1.151 ± 0.020	8.138 ± 0.007	5.853 ± 0.014
Expected Calibration Error (↓ is better) ±95% Confidence Interval					
Pre-Trained	0.163 ± 0.001	0.239 ± 0.001	0.233 ± 0.002	0.287 ± 0.002	0.001 ± 0.000
MAP	0.052 ± 0.001	0.224 ± 0.002	0.060 ± 0.001	0.240 ± 0.002	0.009 ± 0.001
Ensemble	0.153 ± 0.002	0.747 ± 0.004	—	—	0.002 ± 0.000
MC Dropout	0.036 ± 0.000	0.176 ± 0.002	0.054 ± 0.001	0.219 ± 0.002	0.005 ± 0.000
Laplace Diag	0.180 ± 0.002	0.170 ± 0.001	0.072 ± 0.001	0.203 ± 0.001	0.001 ± 0.000
Laplace Full	0.082 ± 0.001	0.136 ± 0.001	0.024 ± 0.000	0.126 ± 0.001	0.000 ± 0.000
RegVar (this paper)	0.019 ± 0.000	0.044 ± 0.001	0.047 ± 0.001	0.024 ± 0.001	0.001 ± 0.000

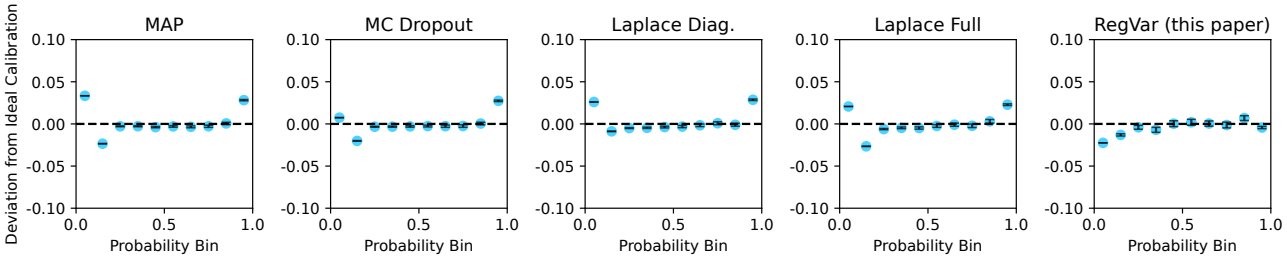


Figure 3: Calibration curves for the methods on the IMDB held-out evaluation with the XL GPT-3 network. Ideal calibration is plotted with dashed lines. 95% confidence intervals are shown with horizontal markers.

RegVar) that considers uncertainty beyond the last layer.

- **Last-Layer Laplace with Diagonal Covariance (Laplace Diag.)** using the inverse of the diagonal of the GGN approximation of the Hessian of the parameters in the last layer of the network.
- **Last-Layer Laplace with Full Covariance (Laplace Full)** using the inverse of the GGN approximation of the Hessian of the parameters in the last layer of the network.
- **Regularization Variation (RegVar)** uses Algorithm 1.

Further variants of the Laplace approximation can be brought to bear on large models (Daxberger et al., 2021). Last-layer diagonal/full covariance Laplace are used here because we found them, in line with previous studies, to be representative of the performance of different Laplace approaches.

Hyperparameters All network weights are regularized by a Gaussian prior with zero mean and low precision (0.02) during fine-tuning. Due to computational constraints, 10 samples are used in the ensemble and MC dropout methods. Dropout rate, prior conditioning matrix for Laplace, and the λ weight in RegVar are tuned to maximize likelihood on validation data. For all methods there is a trade-off between accuracy and calibration; to control for the confounding influence of this trade-off on the metrics, we rescale the predicted variance by a scalar hyperparameter that is chosen for each method to maximize likelihood on the validation datasets.

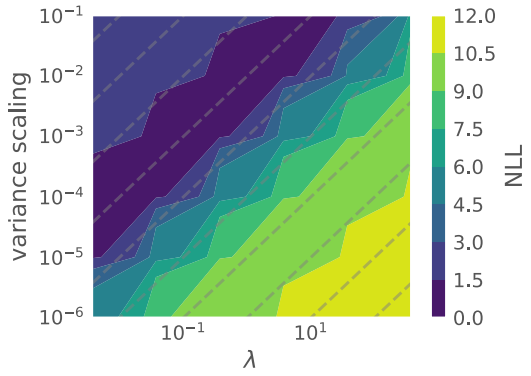


Figure 4: Contour plot of held-out negative log likelihood (↓ is better) for Wikitext validation as function of the λ used to train RegVar and variance rescaling used during inference. Lines isometric to the identity line are shown as gray dashes.

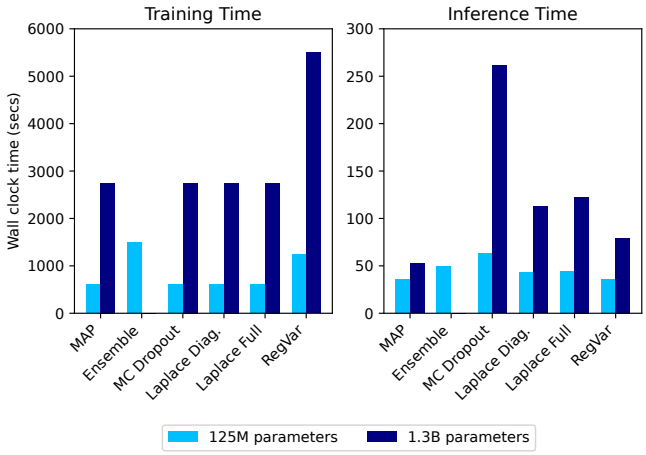


Figure 5: Fine-tune training and inference times for the methods on IMDB dataset. N.B. ensemble was prohibited in the larger model due to memory constraints.

Evaluation All methods are evaluated by held-out negative log likelihood (NLL) and expected calibration error (ECE). NLL penalizes predictive uncertainties that place more probability on the wrong classes, indicating both accuracy and calibration. ECE measures calibration directly using the weighted average error of binned predictions,

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|, \qquad (19)$$

where M is the total number of bins, B_m is the set of samples whose predicted confidence scores fall into the m-th bin, n is the total number of samples, $\text{acc}(B_m)$ is the accuracy of the samples in the m-th bin, and $\text{conf}(B_m)$ is the average predicted confidence of the samples in the m-th bin. We use $M = 10$ throughout. Confidence intervals are calculated using ± 1.96 standard errors (Wald confidence intervals).

For every method, the first step to evaluating the predictive distribution over classes is to estimate the logit means $f_\theta(x_i)$ and variances $\hat{\sigma}_i^2$ (rescaled by the aforementioned hyperparameter). Uncertainty in the logits is then used in the extended probit approximation (Gibbs, 1998) which introduces a logit correction vector κ to account for model variance. The probit approximation correction is given by,

$$\kappa_i = \frac{1}{\sqrt{1 + \frac{\pi \hat{\sigma}_i^2}{8}}}. \qquad (20)$$

The adjusted logits are then used to form predicted probabilities over the clases $p(y_i \mid \mathcal{D}, x_i) \propto \exp\{\kappa_i f_\theta(x_i)\}$.

The implementation was done in PyTorch (Paszke et al., 2017) using HuggingFace[3] to load pre-trained models from `EleutherAI/{gpt-neo-125M,gpt-neo-1.3B}` repositories (Black et al., 2021) originally trained on a 800GB text dataset (Gao et al., 2020) and `google/vit-base-patch16-224-in21k` repositories (Wu et al., 2020) originally trained on ImageNet-21k (Deng et al., 2009). The code to run the experiments can be accessed publicly on GitHub.[4]

Results A comparison of evaluation metrics for the methods is shown in Table 1. We find that RegVar offers significant advantages over MAP across datasets and improves or maintains uncertainty quantification in comparison to the other methods. While Laplace Full (Last Layer) performs robustly across settings, improvements with RegVar highlight the potential benefits of deep uncertainty, i.e., including the sources of variance from the entire network, which is not tractable to target with explicit Hessians at this scale. While NLL measures a combination of accuracy and calibration, ECE isolates the contribution of calibration and allows us to explore how this breaks down further with calibration curves in Figure 3. Overall calibration is good, as evidenced by the small deviation from the ideal line, however most methods tend to over-represent extreme predictions (at 0/1) while RegVar is able to calibrate the predictions away from the extremes.

We explore the empirical behavior of the λ hyperpa-

[3] `huggingface.co`
[4] `https://github.com/jamesmcinerney/regvar`

rameter in RegVar by running fine-tuning with a wide range of different values. Figure 4 shows a contour plot of the effect of λ (x-axis) and the variance rescaling hyperparameter (y-axis), the latter a hyperparameter used in all methods. The isometry of the contours suggests insensitivity of held-out negative log likelihood to the choice of λ during fine-tuning. The two factors appear to cancel each other out and leave a dataset-dependent constant rescaling term (approx. $\frac{1}{100}$ for Wikitext) that does not depend on λ.

A wall clock time comparison for fine-tuning and inference is shown in Figure 5. While inference performance of RegVar is advantageous to all other methods apart from MAP, obtaining the regularized network in RegVar doubles the MAP fine-tuning time, a cost that is amenable to further optimization (e.g. warm starting from MAP) and parallelization in a similar fashion to ensembling and dropout.

Additional experiments are detailed in Appendix C, exploring further settings including using the RegVar framework for parameter uncertainty.

Limitations RegVar is inherently a local method centered around the optimum of the joint distribution and the curvature is evaluated at one location (ignoring other modes). Early stopping during fine-tuning or pre-training also means that the chosen network parameters may not be around the mode of the explicit loss. The finite differences required by RegVar are an additional source of error making it important to select λ small enough, but not so small that underflow errors arise or, more practically, that the noise in the gradients overwhelm the signal from the regularizer.

6 CONCLUSIONS & FUTURE WORK

In this paper, we explored a new concept in Bayesian deep learning that identifies *variation due to regularization* as an approximation to the predictive variance. We considered this phenomenon from a linearized Laplace perspective and evaluated it in large models. There is further work to explore in what cases it is necessary to use the uncertainty of the whole network for predictions to help contextualize the value of RegVar.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful feedback and suggestions.

References

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. `https://doi.org/10.5281/zenodo.5297715`

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. In *International Conference on Machine Learning*. 1613–1622.

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

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

Zhijie Deng, Feng Zhou, and Jun Zhu. 2022. Accelerated linearized Laplace approximation for Bayesian deep learning. *Advances in Neural Information Processing Systems* 35 (2022), 2695–2708.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 2019. 'In-Between' Uncertainty in Bayesian Neural Networks. *arXiv preprint arXiv:1906.11537* (2019).

F Dan Foresee and Martin T Hagan. 1997. Gauss-Newton approximation to Bayesian learning. In *Proceedings of international conference on neural networks (ICNN'97)*, Vol. 3. IEEE, 1930–1935.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027* (2020).

Mark N Gibbs. 1998. *Bayesian Gaussian processes for regression and classification*. Ph.D. Dissertation.

Alex Graves. 2011. Practical variational inference for neural networks. *Advances in neural information processing systems* 24 (2011).

Alexander Immer, Maciej Korzepa, and Matthias Bauer. 2021. Improving predictions of Bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 703–711.

Nathan Kallus and James McInerney. 2022. The implicit delta method. *Advances in Neural Information Processing Systems* 35 (2022), 37471–37483.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems* 30 (2017).

Mohammad Emtiyaz E Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. 2019. Approximate inference turns deep networks into Gaussian processes. *Advances in neural information processing systems* 32 (2019).

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2017).

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

Neil David Lawrence. 2001. *Variational inference in probabilistic models*. Ph.D. Dissertation.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. 2017. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports* 7, 1 (2017), 17816.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. 2020. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1540–1552.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.

David JC MacKay. 1992. Bayesian interpolation. *Neural Computation* 4, 3 (1992), 415–447.

David JC Mackay. 1992. Bayesian methods for adaptive models. *Doctoral Thesis, California Institute of Technology* (1992).

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

Stephan Mandt, Matthew Hoffman, and David Blei. 2016. A variational analysis of stochastic gradient algorithms. In *International conference on machine learning*. PMLR, 354–363.

James Martens and Roger Grosse. 2015. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*. PMLR, 2408–2417.

James Martens and Ilya Sutskever. 2011. Learning recurrent neural networks with Hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 1033–1040.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. arXiv:cs.CL/1609.07843

Luis A Ortega, Simón Rodríguez Santana, and Daniel Hernández-Lobato. 2023. Variational linearized Laplace approximation for Bayesian deep learning. *arXiv preprint arXiv:2302.12565* (2023).

Ian Osband and Benjamin Van Roy. 2015. Bootstrapped Thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300* (2015).

Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, et al. 2024. Position: Bayesian deep learning is needed in the age of large-scale AI. In *International Conference on Machine Learning*. PMLR, 39556–39586.

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

Barak A Pearlmutter. 1994. Fast exact multiplication by the Hessian. *Neural computation* 6, 1 (1994), 147–160.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. A scalable Laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*.

Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Bazan Clement Emile Marcel Raoul, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. 2024. Variational Learning is Effective for Large Deep Networks. In *International conference on machine learning*. PMLR, 44665–44686.

Larry Wasserman. 2006. *All of nonparametric statistics*. Springer.

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

Andrew Gordon Wilson. 2020. The case for Bayesian deep learning. *arXiv preprint arXiv:2001.10995* (2020).

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. 2020. Visual Transformers: Token-based Image Representation and Processing for Computer Vision. arXiv:2006.03677

Adam X. Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. 2024. Bayesian low-rank adaptation for large language models. In *12th International Conference on Learning Representations, ICLR 2024-Conference Track Proceedings*.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence* 40, 6 (2017), 1452–1464.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. Yes
   (c) Clear explanations of any assumptions. Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Yes
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

A PROOFS

We detail here the proofs of the theorems given in the main text. Table 2 summarizes the notation used.

Theorem 4.1 *The derivative of the prediction under the prediction-regularized MAP w.r.t. λ recovers the variance-covariance term in Eq. 7,*

$$\nabla_\lambda f_{\hat{\theta}^{(f(x),\lambda)}}(x)\Big|_{\lambda=0} = \nabla_\theta f_{\hat{\theta}}(x)^\top P^{-1} \nabla_\theta f_{\hat{\theta}}(x), \tag{21}$$

assuming $\hat{\theta}^{(f(x),\lambda)}$ and P^{-1} exist, and that ${\mathcal{L}_\theta}^{(x,\lambda)}$ is continuously differentiable w.r.t. θ.

Proof. To improve readability we use (x, λ) in this proof to refer to fact that evaluation x is passed to the regularizing term (i.e. always $f(x)$). Consider the derivative w.r.t. θ of the prediction-regularized objective at the optimum,

$$\begin{aligned}
&\nabla_\theta {\mathcal{L}_\theta}^{(x,\lambda)}\Big|_{\theta=\hat{\theta}^{(x,\lambda)}} \\
&= \nabla_\theta \ \mathcal{L}_\theta|_{\theta=\hat{\theta}^{(x,\lambda)}} + \lambda \nabla_\theta \ f_\theta(x)|_{\theta=\hat{\theta}^{(x,\lambda)}} \\
&= 0,
\end{aligned} \tag{22}$$

by definition in Eq. 8 and by definition of the optimum. Now, differentiate Eq. 22 w.r.t. λ, applying the chain rule and noticing that $\hat{\theta}^{(x,\lambda)}$ is a function of λ,

$$\begin{aligned}
&\nabla_\theta^2 \ \mathcal{L}_\theta|_{\theta=\hat{\theta}^{(x,\lambda)}} \nabla_\lambda \hat{\theta}^{(x,\lambda)} + \nabla_\theta \ f_\theta(x)|_{\theta=\hat{\theta}^{(x,\lambda)}} \\
&+ \lambda \nabla_\theta^2 \ f_\theta(x)|_{\theta=\hat{\theta}^{(x,\lambda)}} \nabla_\lambda \hat{\theta}^{(x,\lambda)} = 0.
\end{aligned} \tag{23}$$

Evaluating Eq. 23 at $\lambda = 0$ eliminates the term involving the Hessian of the output. Since, by assumption, the inverse Hessian of the log joint exists, rearrange the terms,

$$\nabla_\lambda \ \hat{\theta}^{(x,\lambda)}\Big|_{\lambda=0} = -\left(\nabla_\theta^2 \ \mathcal{L}_\theta|_{\theta=\hat{\theta}^{(x,0)}}\right)^{-1} \nabla_\theta \ f_\theta(x)|_{\theta=\hat{\theta}^{(x,0)}} \tag{24}$$

Finally, substituting Eq. 24 into the chain rule expansion of $\nabla_\lambda \ f_{\hat{\theta}^{(x,\lambda)}}(x)\big|_{\lambda=0}$ yields,

$$\begin{aligned}
&\nabla_\lambda \ f_{\hat{\theta}^{(x,\lambda)}}(x)\big|_{\lambda=0} \\
&= -\nabla_\theta \ f_\theta(x)|_{\theta=\hat{\theta}}^\top \left(\nabla_\theta^2 \ \mathcal{L}_\theta|_{\theta=\hat{\theta}}\right)^{-1} \nabla_\theta \ f_\theta(x)|_{\theta=\hat{\theta}} \,.
\end{aligned} \tag{25}$$

This recovers Eq. 10 by substituting with the definition of P (Eq. 4). □

Theorem 4.2 *The optimal regularization term defined by Eq. 13 is $\hat{r} = \frac{1}{M}\sum_{m=1}^{M} f(\hat{x}_m)$ when $\hat{\theta}^{(f(x),\lambda)}$ and P^{-1} exist and ${\mathcal{L}_\theta}^{(f(x),\lambda)}$ is continuously differentiable w.r.t. θ.*

Proof. Take the derivative of the objective in Eq. 13 w.r.t. r,

$$\begin{aligned}
&\nabla_r \sum_{m=1}^{M} \|\hat{\theta}^{(f(x_m),\lambda)} - \hat{\theta}^{(r,\lambda)}\|_2^2 \\
&= \nabla_r \sum_{m=1}^{M} \|(\hat{\theta}^{(f(x_m),\lambda)} - \hat{\theta}) - (\hat{\theta}^{(r,\lambda)} - \hat{\theta}))\|_2^2 \\
&= \nabla_r \sum_{m=1}^{M} \|P^{-1}\nabla_\theta f(\hat{x}_m) - P^{-1}\nabla_\theta r\|_2^2 \qquad (26) \\
&= \sum_{m=1}^{M} \nabla_r \|P^{-1}(\nabla_\theta f(\hat{x}_m) - \nabla_\theta r)\|_2^2 \\
&= 2(P^{-1})^\top P^{-1}(\sum_{m=1}^{M}(\nabla_\theta f(x_m) - \nabla_\theta r)), \qquad (27)
\end{aligned}$$

Table 2: **Notation Summary**

Symbol	Definition
θ	Network parameters.
$\mathcal{L}_\theta$	Log joint distribution parameterized by θ.
$\hat{\theta}$	Maximum *a posteriori* parameters under the joint distribution (Eq. 1).
$f_{\hat{\theta}}$	Network output with parameters $\hat{\theta}$.
$\nabla_\theta f_{\hat{\theta}}$	Derivative of network output w.r.t. θ evaluated at $\theta = \hat{\theta}$.
${\mathcal{L}_\theta}^{(t_\theta,\lambda)}$	Log joint distribution parameterized by θ and regularized by λt_θ, e.g. t_θ could be a network output $f_\theta(x)$ (Eq. 8) or fine-tuning regularizer r (Eq. 13).
$\hat{\theta}^{(t_\theta,\lambda)}$	Parameters that maximize ${\mathcal{L}_\theta}^{(t_\theta,\lambda)}$, e.g., see Eq. 9.
$\nabla_\lambda f_{\hat{\theta}^{(t_\theta,\lambda)}}$	Derivative of network output under parameters $\hat{\theta}^{(t_\theta,\lambda)}$ w.r.t. regularization weight λ. Note that the optimal network parameters themselves are a function of λ, making the overall term a non-linear expression involving $\nabla_\lambda f_{\hat{\theta}}$ (see Eq. 21).

using Theorem 4.1 to express the difference in optimal parameters in terms involving the inverse Hessian in Eq. 26. Finally, set the derivative in Eq. 27 to zero and solve for r, the unique minimizer of the objective (up to a constant that does not depend on θ). □

B ALGORITHMS FOR REGULARIZATION VARIATION (REGVAR)

In this section we expand on variations of the REGVAR strategy and provide relevant algorithms where applicable.

B.1 Pointwise REGVAR

The algorithm that follows most directly from Theorem 1 targets the uncertainty calculation for an arbitrary evaluation point x (not necessarily from the training or test set) and is given in Algorithm 2. The computational burden of recalculating the optimal regularized parameters for every different evaluation input means that this approach is not scalable to large models, motivating extensions to the regularization term which we consider next.

Algorithm 2 Pointwise REGVAR

Input: network f_θ, fitted MAP $\hat{\theta}$, training data $\mathcal{D}$ of size n, evaluation input x, scalar λ, step size schedule $\gamma(\cdot)$
Warm start $\hat{\theta}^{(x,\lambda)} \leftarrow \hat{\theta}$
Initialize step counter $j = 0$
repeat
  Sample training example $(x_i, y_i) \sim \mathcal{D}$
  Follow stochastic gradient $\hat{\theta}^{(x,\lambda)} \leftarrow \hat{\theta}^{(x,\lambda)} + N\gamma(j)\nabla_\theta \tilde{\mathcal{L}}_\theta^{(x,\lambda)}(x_i, y_i)$
  $j \leftarrow j + 1$
until $\hat{\theta}^{(x,\lambda)}$ converges
Return: predictive variance $\hat{\sigma}_f^2(x) := \frac{1}{\lambda}|f_{\hat{\theta}^{(x,\lambda)}}(x) - f_{\hat{\theta}}(x)|$

B.2 Data-Augmentation Objective

When the likelihood is Gaussian, the REGVAR regularization term can be absorbed into the likelihood. In particular, completing the square in Eq. 16, by pulling the term $\frac{\lambda\sigma^2}{nM}(\frac{\lambda\sigma^2}{2nM} - y_i)$ from a constant that does not

Algorithm 3 Amortized RegVar for Parameter Uncertainty

Input: network f_θ, fitted MAP $\hat{\theta}$, training data $\mathcal{D}$ of size n, evaluation inputs $\{\hat{x}_{i=1}^m\}$, scalar λ, step size schedule $\gamma(\cdot)$
Warm start $\hat{\theta}^{(\hat{r},\lambda)} \leftarrow \hat{\theta}$
Initialize step counter $j = 0$
repeat
  Sample training example $(x_i, y_i) \sim \mathcal{D}$
  Follow gradient $\hat{\theta}^{(\hat{r},\lambda)} \leftarrow \hat{\theta}^{(\hat{r},\lambda)} + n\gamma(j)\nabla_\theta \mathcal{L}_i^{(\text{PU})}$ `//See Eq. 18`
  $j \leftarrow j + 1$
until $\hat{\theta}^{(\hat{r},\lambda)}$ converges
Return: parameter variance $\hat{\sigma}_\theta^2(x) := \frac{1}{\lambda}|\hat{\theta}^{(\hat{r},\lambda)} - \hat{\theta}|$

depend on θ, yields the objective,

$$\mathcal{L}_i^{(\text{DA})} = \log \mathcal{N}\left(y_i + \frac{\tilde{\lambda}}{nM}\sigma^2 \mid f_\theta(x_i), \sigma^2\right) + \frac{1}{n}\log p(\theta), \tag{28}$$

for observation variance hyperparameter σ^2. Under the common Gaussian likelihood assumption, Eq. 28 implies that pre-trained RegVar is performed by data augmentation, without requiring *any* adjustment to the model architecture or training procedure.

B.3 RegVar for Parameter Uncertainty

For completeness, we present Algorithm 3 that corresponds to the parameter uncertainty objective given in Eq.18 in the main text. We apply this algorithm in Section C.2.

C FURTHER EXPERIMENTS AND HYPERPARAMETER SELECTION

In this section, we evaluate RegVar and compare against benchmarks and the full Hessian (where possible).

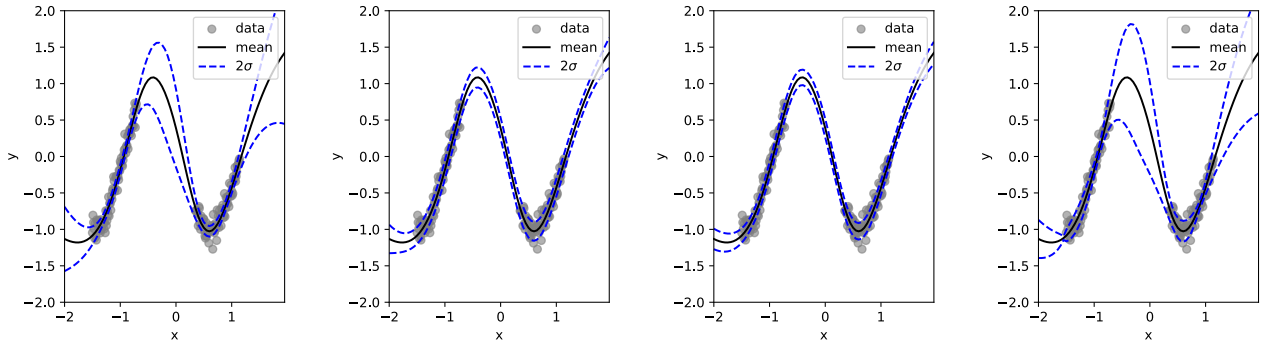
C.1 Synthetic Data

We explore four data generating distributions designed to test both in-distribution inference and extrapolation to "in-between" uncertainty (Foong et al., 2019). A feedforward neural network is used with 50 hidden units and tanh activations to ensure that the Hessian exists, and optimized using Adam (Kingma and Ba, 2017) with a learning rate of 0.005. Four methods for quantifying the epistemic uncertainty of the network are compared,

- **Exact Hessian** using Eq. 4 and both terms in the likelihood Hessian (Eq. 5).
- **GGN** using Eq. 4 with only the second term in likelihood Hessian (Eq. 5).
- **Eigenvector approximation** of the GGN for $P^{-1} \approx A\Lambda^{-1}A^\top$ using the top k eigenvectors A of P (with corresponding eigenvalues Λ) where $k = \log_e(\dim(\theta))$ rounded to the nearest positive integer. The GGN is chosen because it only makes sense to use the eigenvector approximation when the Hessian is large.
- **RegVar** (this paper), using Algorithm 1.

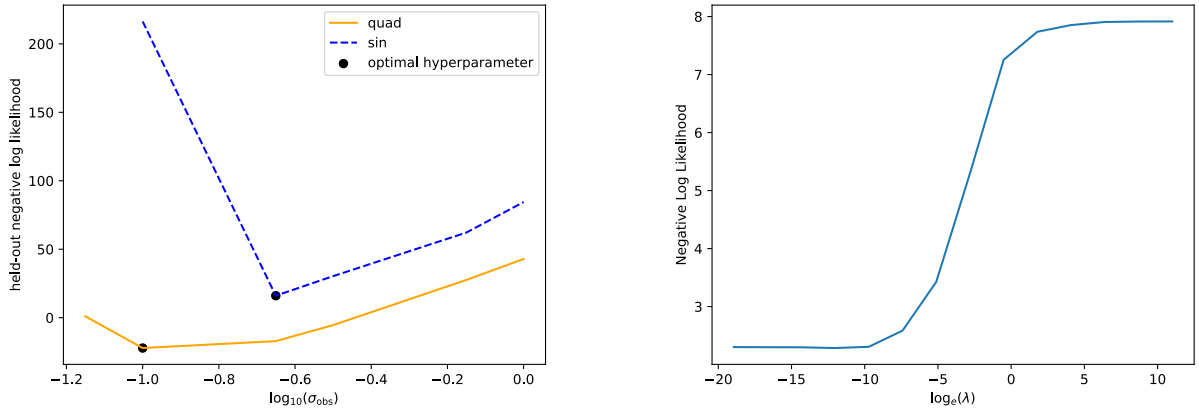
The synthetic data-generating distributions are,

- **Quadratic-Uniform** Draw 32 data points $x \sim \mathcal{N}(0, 1)$ and let $y = \frac{1}{10}x^2 - \frac{1}{2}x + 5 + \frac{1}{10}\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$.
- **Quadratic-Inbetween** Draw 16 data points $x \sim \text{Uniform}(-2, -\frac{1}{2})$ and 16 data points $x \sim \text{Uniform}(\frac{4}{5}, \frac{5}{2})$, using the same response distribution as Quadratic-Uniform.
- **Sin-Uniform** Sample 160 data points $x \sim \text{Uniform}(-\frac{3}{2}, \frac{23}{20})$ and let $y = -\sin(3x - \frac{3}{10}) + \frac{1}{10}\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$.



(a) Full Hessian (b) GGN (c) Eigen. Approx. (d) HFL (this paper)

Figure 6: Illustration of epistemic uncertainties of the predicted mean for the Sin-Inbetween OOD task.



(a) Hyperparameter sweeps for observational variance. (b) Lambda sweep

Figure 7: Hyperparameter selection for modeling and RegVar algorithm.

- **Sin-Inbetween** Fix 160 data points evenly spaced in ranges $[-1.5, -0.7)$ and $[0.35, 1.15)$ and let use the same response as Sin-Uniform.

For both the quadratic and sin datasets, the observational variance hyperparameter was selected through grid search over the set $\{0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$ minimizing negative log likelihood for a held-out validation set of equal size to the training dataset. The results are summarized in Figure 7(a), with the optimal hyperparameter chosen indicated by a marker. The prior variance was fixed at 3.0, which was the minimum value to the nearest 0.1 for which the Hessian was sufficiently conditioned across all tasks to allow for inversion.

C.2 Parameter Uncertainty Experiment

We also evaluated the parameter regularization approach given in Eq. 18. Figure 9 shows the 1 standard deviation interval about the predicted mean under the various methods. Recall that RegVar for parameter uncertainty requires only to apply a small amount of additional L1 regularization to the parameters. Nonetheless, the intervals indicated often align with those requiring full Hessian inversion. These parameter uncertainties may be used for model introspection. As an example, we considered the task of adding sparsity to a network, setting parameters to 0 when their uncertainty intervals overlap with 0. The mean absolute errors of the resulting predictions are shown in Figure 8. RegVar does competitively, particularly for in-between uncertainty. An outlier result is the absolute error of the full Hessian in the Quadratic-Inbetween in-distribution task. This was due to mis-selecting an entire group of units to be set to 0, greatly impacting the shape of predictions.

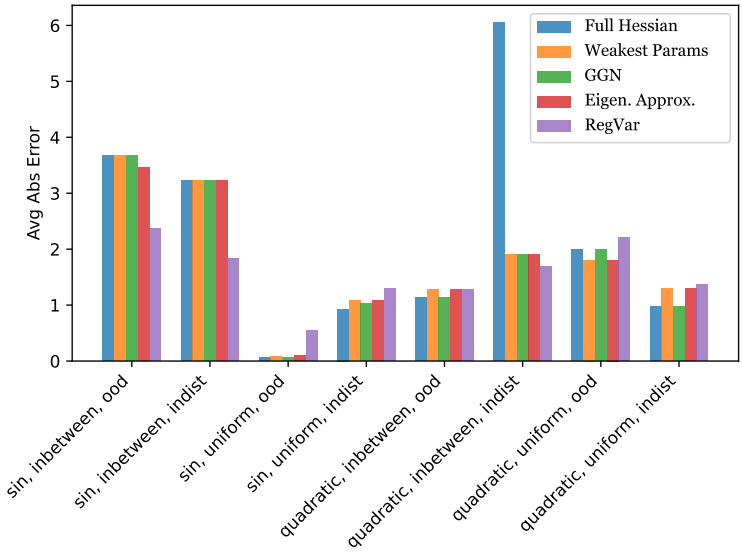


Figure 8: Absolute errors in predicted means for all the tasks with networks with added sparsity using the uncertainty quantification methods.

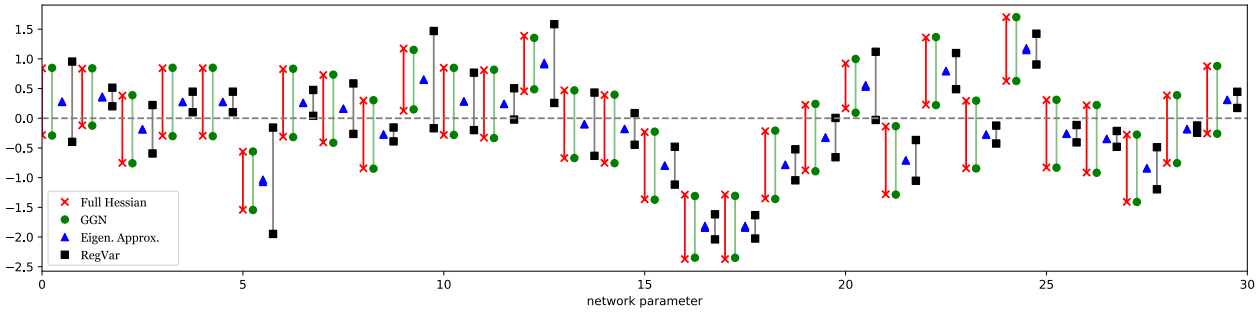


Figure 9: The uncertainty ranges of a random subset of 30 parameters of the neural network indicate that many parameters include 0 in their intervals. RegVar recovers much of the structure of these uncertainties via additional regularization of the network parameters in the objective function.