

# BAYESIAN PREDICTIVE CODING

**Alexander Tschantz<sup>1,2</sup> Magnus Koudahl<sup>1</sup> Hampus Linander<sup>1</sup> Lancelot Da Costa<sup>1</sup>**  
**Conor Heins<sup>1,3</sup> Jeff Beck<sup>4</sup> Christopher Buckley<sup>1,2</sup>**

<sup>1</sup>VERSES AI Research Lab, Los Angeles, CA, USA

<sup>2</sup>School of Engineering and Informatics, University of Sussex, Brighton, UK

<sup>3</sup>Max Planck Institute of Animal Behavior, Department of Collective Behaviour, Konstanz, Germany

<sup>4</sup>Department of Neurobiology, Duke University, Durham, NC, USA

{alec.tschantz, magnus.koudahl, hampus.linander, lance.dacosta,  
conor.heins, christopher.buckley}@verses.ai, jeff.beck@duke.edu

## Abstract

Predictive coding (PC) is an influential theory of information processing in the brain, providing a biologically plausible alternative to backpropagation. It is motivated in terms of Bayesian inference, as hidden states and parameters are optimised via gradient descent on variational free energy. However, implementations of PC rely on maximum *a posteriori* (MAP) estimates of hidden states and maximum likelihood (ML) estimates of parameters, limiting their ability to quantify epistemic uncertainty. In this work, we investigate a Bayesian extension to PC that estimates a posterior distribution over network parameters. This approach, termed Bayesian Predictive coding (BPC), preserves the locality of PC and results in closed-form Hebbian weight updates. Compared to PC, our BPC algorithm converges in fewer epochs in the full-batch setting and remains competitive in the mini-batch setting. Additionally, we demonstrate that BPC offers uncertainty quantification comparable to existing methods in Bayesian deep learning, while also improving convergence properties. Together, these results suggest that BPC provides a biologically plausible method for Bayesian learning in the brain, as well as an attractive approach to uncertainty quantification in deep learning.

## 1 Introduction

Originating in the domain of neuroscience, the predictive coding (PC) framework [1, 2, 3, 4] proposes that the function of neural plasticity is to minimize local *prediction errors*, which quantify the difference between estimated and observed signals. This framework has been adapted as a method for training deep neural networks using only local information, offering a biologically plausible alternative to backpropagation (BP) [5, 6, 7, 8]. PC offers several benefits over BP, including improved performance in online and continual learning settings [9], favourable optimization properties [10], the flexibility to be used in either a generative or discriminative manner [11] and intrinsic auto-associative memory capabilities [12].

PC has traditionally been motivated through variational Bayesian inference [13, 14], characterising the relationships between hidden states and parameters using probability distributions. However, implementations of PC generally do not operate on distributions but instead use maximum *a posteriori* (MAP) estimates of hidden states and maximum likelihood (ML) estimates of parameters. This contrasts with Bayesian deep learning [15], where the goal is to estimate posterior distributions over parameters so that epistemic and aleatoric uncertainty [16] can be quantified and subsequently used for model comparison, network pruning, or well-calibrated confidence estimation. Regardless,

uncertainty quantification is essential for the robustness, reliability, and interpretability of learning systems, and how it is performed by the brain remains an open question.

In this work, we propose an extension of PC that estimates approximate Bayesian posteriors over network parameters. This approach, termed Bayesian predictive coding (BPC), parameterises neural activity in a way that allows for the use of conjugate priors, enabling closed-form update rules for the weight distribution. The resulting updates are Hebbian functions of the pre- and post-synaptic activity, and hidden state updates retain their interpretation as precision-weighted prediction errors, thus preserving the locality and simplicity of the PC algorithm. Moreover, the ability to compute posterior updates in a closed form means that BPC can converge in substantially fewer iterations than gradient-based methods [17]. See Appendix E for an overview of related work.

In a set of experiments, we empirically verify that BPC achieves comparable performance to PC and traditional BP in full-batch training and remains competitive in mini-batch training. Notably, in the full-batch training context, BPC converges in a remarkably few epochs. Moreover, we demonstrate that the learned posterior distribution enables robust quantification of epistemic and aleatoric uncertainties in synthetic regression tasks. We compare BPC to a popular benchmark in Bayesian deep learning and demonstrate that our method provides both improved uncertainty quantification and improved accuracy and converges in fewer iterations. Taken together, our results suggest that BPC is a viable method for training uncertainty-aware neural networks using local information, suggesting a potential mechanism for uncertainty quantification in neural systems.

## 2 Methods

Bayesian predictive coding is an algorithm for inverting hierarchical Gaussian generative models with  $L$  layers of variables  $\mathbf{Z} = \{\mathbf{z}_l : l = 0 \dots L\}$  and parameters  $\Theta = \{\mathbf{W}_l, \Sigma_l : l = 1 \dots L\}$ :

$$\begin{aligned} p(\mathbf{Z}, \Theta) &= p(\mathbf{z}_0)p(\mathbf{W}_0, \Sigma_0) \prod_{l=1}^L p(\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{W}_l, \Sigma_l)p(\mathbf{W}_l, \Sigma_l), \\ p(\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{W}_l, \Sigma_l) &= \mathcal{N}(\mathbf{z}_l | \mathbf{W}_l f(\mathbf{z}_{l-1}), \Sigma_l), \\ p(\mathbf{W}_l, \Sigma_l) &= \mathcal{MN}(\mathbf{W}_l | \mathbf{M}_l^{(0)}, \Sigma_l^{-1}, \mathbf{V}_l^{(0)}) \mathcal{W}(\Sigma_l^{-1} | \boldsymbol{\Psi}_l^{(0)}, \nu_l^{(0)}), \end{aligned} \quad (1)$$

where  $f(\cdot)$  is a point-wise non-linear activation function, and  $p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0 | \mu_0, \Sigma_0)$ . Notably, we have placed the parameters  $\mathbf{W}_l$  outside of the non-linear activation function  $f(\cdot)$ , which is essential for enabling the closed-form updates used in Equation 7. In practice, we also use a bias term that we ignore here for simplicity. Given that the log-likelihood is quadratic in the network parameters, a Matrix Normal Wishart prior for parameters  $\mathbf{W}_l$  and  $\Sigma_l$  is conditionally conjugate given network activity  $\mathbf{Z}$ . See Appendix A for relevant definitions.

In the standard PC framework, latent variables  $\mathbf{Z}$  are represented via maximum a posteriori (MAP) estimates, and parameters  $\Theta$  are represented via maximum likelihood (ML) estimates. In contrast, BPC augments this approach by representing an approximate posterior distribution over parameters,

$$\begin{aligned} q_\lambda(\Theta) &= \prod_{l=1}^L q(\mathbf{W}_l, \Sigma_l), \\ q_\lambda(\mathbf{W}_l, \Sigma_l) &= \mathcal{MN}(\mathbf{W}_l | \mathbf{M}_l, \Sigma_l^{-1}, \mathbf{V}_l) \mathcal{W}(\Sigma_l^{-1} | \boldsymbol{\Psi}_l, \nu_l). \end{aligned} \quad (2)$$

where  $\lambda = \{\mathbf{M}_l, \mathbf{V}_l, \boldsymbol{\Psi}_l, \nu_l : l = 1 \dots L\}$  are the variational parameters. Given Equation (1) and (2), we define the objective function  $\mathcal{E}(\mathbf{Z}, \lambda)$  as:

$$\mathcal{E}(\mathbf{Z}, \lambda) = \left\langle \log q_\lambda(\Theta) - \log p(\mathbf{Z}, \Theta) \right\rangle_{q_\lambda(\Theta)} \quad (3)$$

which is equivalent to the variational free energy under the assumption that  $q_\lambda(\mathbf{Z})$  is a Dirac delta distribution [13]. See Appendix C comparing Equation (3) to the energy function used in PC. To estimate  $\mathbf{Z}$  and  $\Theta$ , we use the expectation-maximisation (EM) algorithm [18] on  $\mathcal{E}(\mathbf{Z}, \lambda)$ :

$$\mathbf{Z}^* = \arg \min_{\mathbf{Z}} \mathcal{E}(\mathbf{Z}, \lambda), \quad \lambda^* = \arg \min_{\lambda} \mathcal{E}(\mathbf{Z}^*, \lambda). \quad (4)$$

Following PC, we solve the first optimisation problem using gradient descent on the energy function  $\mathcal{E}(\mathbf{Z}, \boldsymbol{\lambda})$ , which can be rewritten as:

$$\mathcal{E}(\mathbf{Z}, \boldsymbol{\lambda}) = \frac{1}{2} \sum_{l=1}^L \underbrace{\left\langle (\mathbf{z}_l - \mathbf{W}_l f(\mathbf{z}_{l-1}))^\top \Sigma_l^{-1} (\mathbf{z}_l - \mathbf{W}_l f(\mathbf{z}_{l-1})) \right\rangle}_{\mathcal{E}_l}_{q_{\boldsymbol{\lambda}}(\mathbf{W}_l, \Sigma_l)} + C, \quad (5)$$

where  $C$  represents terms independent of  $\mathbf{Z}$  and  $\mathcal{E}_l$  is the *precision-weighted prediction error* for layer  $l$ . Gradient descent is performed by iteratively updating the latent variables  $\mathbf{Z}$  using the following update rule:

$$\mathbf{z}_l \leftarrow \mathbf{z}_l - \alpha \frac{\partial \mathcal{E}}{\partial \mathbf{z}_l}, \quad \frac{\partial \mathcal{E}}{\partial \mathbf{z}_l} = \frac{1}{2} \left( \frac{\partial \mathcal{E}_l}{\partial \mathbf{z}_l} + \frac{\partial \mathcal{E}_{l+1}}{\partial \mathbf{z}_l} \right), \quad (6)$$

where  $\alpha$  is the learning rate. This process is repeated until convergence or a maximum number of iterations  $T$ , and the converged latent variables are denoted as  $\mathbf{Z}^*$ . See Appendix B for the closed-form expressions of the derivatives in Equation (6) as well as a method for optimally selecting  $\alpha$ .

Given the converged latent variables  $\mathbf{Z}^*$ , we can find a closed-form expression for  $\arg \min_{\boldsymbol{\lambda}} \mathcal{E}(\mathbf{Z}^*, \boldsymbol{\lambda})$ . Specifically, we update the natural parameters  $\boldsymbol{\eta}_l$  of the variational posterior  $q_{\boldsymbol{\lambda}}(\mathbf{W}_l, \Sigma_l)$ :

$$\boldsymbol{\eta}_l^* = \boldsymbol{\eta}_l^{(0)} + \sum_n \left( f(\mathbf{z}_{l-1}^{*n}) f(\mathbf{z}_{l-1}^{*n})^\top, f(\mathbf{z}_{l-1}^{*n}) \mathbf{z}_l^{*n\top}, \mathbf{z}_l^{*n} \mathbf{z}_l^{*n\top}, \mathbf{1} \right), \quad (7)$$

where  $\boldsymbol{\eta}_l^{(0)}$  are the natural parameters for the prior distribution defined in Equation (1), and  $\mathbf{1}$  denotes a vector of ones whose dimension matches the number of data points  $n$ . Equation 7 has a natural interpretation as a Hebbian function of pre- and post-synaptic activity. See Appendix D for a proof that Equation 7 minimises  $\mathcal{E}(\mathbf{Z}^*, \boldsymbol{\lambda})$ . The natural parameters  $\boldsymbol{\eta}_l$  relate to the parameters of Equation (2) via:

$$\boldsymbol{\eta}_l = (\mathbf{V}_l^{-1}, \mathbf{M}_l \mathbf{V}_l^{-1}, \boldsymbol{\Psi}_l^{-1} + \mathbf{M}_l \mathbf{V}_l^{-1} \mathbf{M}_l^\top, \nu_l - d_{y_l} + d_{x_l} - 1). \quad (8)$$

Equation 7 provides an exact solution to the optimisation problem when the latent variables  $\mathbf{Z}^*$  are estimated for the entire dataset. In practice, mini-batching is often used, in which case we introduce a learning rate  $\kappa$  and update the natural parameters as  $\boldsymbol{\eta}_l^* = (1 - \kappa)\boldsymbol{\eta}_l + \kappa\boldsymbol{\eta}_l^*$ . This minibatch update rule corresponds to performing stochastic natural gradient descent on the variational parameters, leveraging the geometry induced by the variational posterior distribution [19].

---

#### Algorithm 1 Bayesian Predictive Coding

---

```

1: Randomly initialize  $\boldsymbol{\eta}$  and set prior parameters  $\boldsymbol{\eta}^{(0)} = (\mathbf{M}^{(0)} \mathbf{V}^{(0)} \boldsymbol{\Psi}^{(0)}, \nu^{(0)})$ 
2: for each  $(\mathbf{x}, \mathbf{y})$  batch do
3:   repeat
4:     Initialize  $\mathbf{Z}$ 
5:     for  $l = 1$  to  $L$  do
6:        $\mathbf{z}_l \leftarrow \mathbf{z}_l - \alpha \frac{\partial \mathcal{E}}{\partial \mathbf{z}_l}$ 
7:     end for
8:     until convergence of  $\mathbf{z}$  or maximum  $T$  iterations
9:   for  $l = 1$  to  $L$  do
10:     $\boldsymbol{\eta}_l \leftarrow \boldsymbol{\eta}_l^{(0)} + \sum_n \left( f(\mathbf{z}_{l-1}^{*n}) f(\mathbf{z}_{l-1}^{*n})^\top, f(\mathbf{z}_{l-1}^{*n}) \mathbf{z}_l^{*n\top}, \mathbf{z}_l^{*n} \mathbf{z}_l^{*n\top}, \mathbf{1} \right)$ 
11:   end for
12: end for

```

---

**Training and testing** Given a dataset  $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ , Bayesian Predictive Coding (BPC) trains models in a discriminative manner by fixing the input nodes to  $\mathbf{z}_0 = \mathbf{x}^{(i)}$  and the output nodes to  $\mathbf{z}_L = \mathbf{y}^{(i)}$ . Alternatively, the model can also be trained in an unsupervised setting by only fixing the top layer  $\mathbf{z}_L$ . For each mini-batch of data, we iteratively apply Equation (4), as detailed in Algorithm 1.

During testing, we handle the uncertainty captured by the parameter posterior  $q_{\boldsymbol{\lambda}}(\boldsymbol{\Theta})$  in three distinct ways:

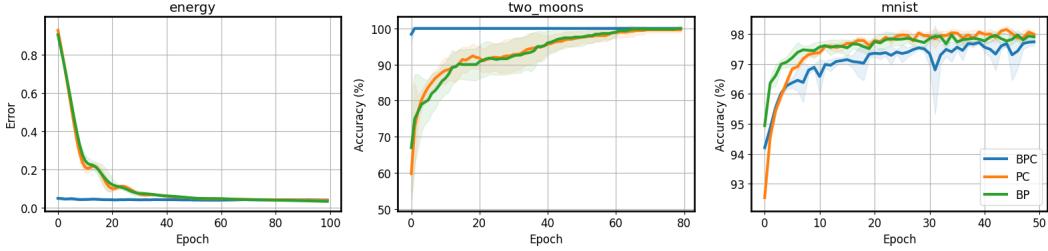


Figure 1: Accuracy on two classification tasks (`two_moons` and MNIST) and error on one regression task (`energy`). These results show that BPC converges to the same accuracy in fewer epochs in the full-batch setting (`energy` and `two_moons`), and is competitive in the mini-batch setting (MNIST). Shaded regions denote 1 standard deviation across 5 seeds. See Appendix F for experiment details.

**Deterministic forward pass:** We use the expected parameter values at each layer, effectively ignoring uncertainty:

$$\langle \mathbf{W}_l, \Sigma_l \rangle_{q_\lambda(\mathbf{W}_l, \Sigma_l)} . \quad (9)$$

**Monte Carlo sampling:** We sample parameters from their posterior distributions:

$$\mathbf{W}_l \sim \mathcal{MN}(\mathbf{W}_l | \mathbf{M}_l, \Sigma_l^{-1}, \mathbf{V}_l), \quad \Sigma_l^{-1} \sim \mathcal{W}(\Sigma_l^{-1} | \Psi_l, \nu_l), \quad (10)$$

and average the network outputs across multiple samples to estimate predictive uncertainty.

**Analytical uncertainty propagation:** We analytically propagate the uncertainty through each layer of the network by computing:

$$q(\mathbf{z}_l) \propto \left\langle \mathcal{N}(\mathbf{z}_l | \mathbf{W}_l \mathbf{z}_{l-1}, \Sigma_l) \right\rangle_{q(\mathbf{z}_{l-1}) q_\lambda(\mathbf{W}_l, \Sigma_l)}, \quad (11)$$

where  $q(\mathbf{z}_{l-1})$  is approximated as Gaussian (or Dirac delta for the input layer). The uncertainty is propagated through the nonlinear activation function  $f(\cdot)$  using the deterministic approximation method from [20], suitable for the ReLU activations employed in our experiments.

### 3 Experiments

#### 3.1 Accuracy

We implement the BPC algorithm to train ReLU networks and evaluate their accuracy to networks trained via PC and BP. Specifically, we examine performance on two small datasets (the `energy` dataset from the UCI repository [21] and the `two_moons` dataset [22]) using full-batch training, and one larger dataset (MNIST [23]) using mini-batch training. See Appendix F for details on hyperparameters and dataset specifics.

The results of these experiments are shown in Figure 1. In the full-batch setting, BPC converges in the first few epochs, since it utilises closed-form updates for the parameter posterior, whereas PC and BP both take several epochs to converge. In the mini-batch setting, BPC is competitive with both PC and BP, converging to an accuracy that is, on average, within 0.3% of these methods. We note that BP and PC are both optimised using the Adam optimiser [24], with PC additionally requiring weight decay [25]. When trained with vanilla stochastic gradient descent (SGD), BP and PC converge significantly slower than BPC, and the accuracy of PC is often significantly lower. These experiments confirm that the posterior updates in Equation 7 provide a viable method for training Bayesian deep neural networks using local update rules.

#### 3.2 Uncertainty quantification

To evaluate the learned posterior  $q_\lambda(\Theta)$ , we train a compact BPC model on two synthetic regression tasks and empirically verify that the model can quantify aleatoric and epistemic uncertainty. To quantify aleatoric uncertainty, we propagate uncertainty through the network to estimate the first

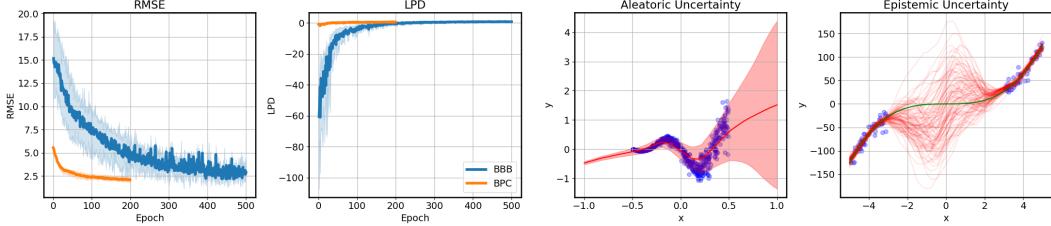


Figure 2: Comparison of root mean square error (RMSE) and log predictive density (LPD) metrics on the UCI regression dataset *yacht* using Bayesian Predictive Coding (BPC) and Bayes by Backprop (BBB). The final two plots show the aleatoric and epistemic uncertainties quantified by the model on synthetic regression tasks. See Appendix F for experiment details.

and second-order moments of the output. This approach naturally accommodates homoscedastic variance; for heteroscedastic variance, we additionally parameterise the output layer with a variance node following the method described in [20]. We quantify epistemic uncertainty by drawing multiple samples from the parameter posterior  $q_{\Theta}(\Theta)$  and visualizing the predicted functions. These results, illustrated in Figure 2, confirm that our model accurately captures both forms of uncertainty. See Appendix F for details on the regression tasks and network hyperparameters used.

Finally, we compare BPC to a popular Bayesian deep learning benchmark, *Bayes by Backprop* (BBB) [26]. BBB estimates variational free energy by drawing samples from the posterior distribution and subsequently uses backpropagation to update the variational parameters. We compare BPC and BBB using log predictive density (LPD) and root mean squared error (RMSE) across several UCI regression tasks [21]. For both methods, LPD is calculated by drawing multiple samples from the posterior distribution over weights and computing the average log-likelihood of the data points. Table 1 shows that BPC outperforms BBB in terms of both LPD and RMSE on most tasks. Figure 2 plots LPD and RMSE against training epochs for the *yacht* dataset, demonstrating faster convergence of BPC due to its closed-form updates. We observe similar improvements in convergence across other datasets.

Dataset	LPD		RMSE	
	BBB	BPC	BBB	BPC
<i>yacht</i>	<b>1.02</b>	0.75	2.16	<b>2.08</b>
<i>concrete</i>	-0.77	<b>-0.59</b>	6.03	<b>5.60</b>
<i>wine</i>	-8.93	<b>-2.49</b>	0.64	<b>0.60</b>
<i>housing</i>	-0.56	<b>-0.49</b>	2.81	<b>2.62</b>
<i>power</i>	-0.39	<b>-0.14</b>	<b>4.11</b>	4.13
<i>energy</i>	<b>0.33</b>	0.28	1.86	<b>1.51</b>

Table 1: Comparison between Bayesian Predictive Coding (BPC) and Bayes by backprop (BBB) in terms of the average log-likelihood (LPD) and root mean square error (RMSE) for UCI regression datasets. See Appendix F for experiment details.

## 4 Discussion

In this work, we have introduced Bayesian predictive coding (BPC), an algorithm that extends predictive coding (PC) by incorporating Bayesian posterior distributions over model parameters. We demonstrated that the resulting update rules naturally translate into Hebbian functions of pre- and post-synaptic activity, thus preserving the local computations and biological plausibility central to PC. Additionally, these update rules provide closed-form expressions for posterior parameters, enabling enhanced convergence properties in full-batch training. Collectively, our findings suggest that BPC offers a viable approach for implementing Bayesian neural networks in biological systems.

The current work has two main limitations. First, it inherits the computational cost associated with performing gradient descent on latent variables  $Z$  before each weight update. This limitation is

also shared by PC, resulting in similar computational times per epoch for both algorithms. Second, the use of the Matrix Normal Wishart distribution for parameter posteriors introduces additional computational complexity. For larger neural networks, structured low-rank approximations will become necessary, and the choice of posterior approximations is an important direction for future research. We note that all experiments presented here were executed on consumer-grade CPUs.

Several avenues for further exploration remain. For instance, it would be possible to take a model pre-trained using backpropagation and subsequently apply BPC to quantify model uncertainty on new data batches by estimating latent variables  $\tilde{\mathbf{Z}}$  and performing closed-form posterior updates for  $\lambda$ . Additionally, further research could investigate the optimization properties of BPC more thoroughly. For example, the current estimate of  $\Sigma$  acts as an adaptive learning rate during inference of latent variables  $\mathbf{Z}$ , dynamically emphasizing more informative (low-variance) dimensions. Furthermore, employing conjugate priors  $p(\Theta)$  may promote beneficial optimization behavior; for instance, favoring identity-like priors for  $\Sigma$  may encourage independence among latent dimensions, potentially leading to more disentangled representations conducive to cross-task generalization.

## References

- [1] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.
- [2] David Mumford. On the computational architecture of the neocortex: II the role of cortico-cortical loops. *Biological cybernetics*, 66(3):241–251, 1992.
- [3] Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.
- [4] Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021.
- [5] James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.
- [6] Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022.
- [7] Tommaso Salvatori, Yuhang Song, Zhenghua Xu, Thomas Lukasiewicz, and Rafal Bogacz. Reverse differentiation via predictive coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8150–8158, 2022.
- [8] Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566–22579, 2020.
- [9] Yuhang Song, Beren Millidge, Tommaso Salvatori, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature neuroscience*, 27(2):348–358, 2024.
- [10] Francesco Innocenti, El Mehdi Achour, Ryan Singh, and Christopher L Buckley. Only strict saddles in the energy landscape of predictive coding networks? *arXiv preprint arXiv:2408.11979*, 2024.
- [11] Beren Millidge, Tommaso Salvatori, Yuhang Song, Rafal Bogacz, and Thomas Lukasiewicz. Predictive coding: Towards a future of deep learning beyond backpropagation? *arXiv preprint arXiv:2202.09467*, 2022.
- [12] Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34:3874–3886, 2021.
- [13] Christopher L Buckley, Chang Sub Kim, Simon McGregor, and Anil K Seth. The free energy principle for action and perception: A mathematical review. *Journal of mathematical psychology*, 81:55–79, 2017.
- [14] Rafal Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76:198–211, 2017.
- [15] Theodore Papamarkou, Maria Skoulikidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, et al. Position: Bayesian deep learning is needed in the age of large-scale ai. In *Forty-first International Conference on Machine Learning*, 2024.
- [16] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.
- [17] Conor Heins, Hao Wu, Dimitrije Markovic, Alexander Tschantz, Jeff Beck, and Christopher Buckley. Gradient-free variational learning with conditional mixture networks. *arXiv preprint arXiv:2408.16429*, 2024.

- [18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.
- [19] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.
- [20] Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, Jose Miguel Hernandez-Lobato, and Alexander L Gaunt. Deterministic variational inference for robust bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.
- [21] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- [22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [23] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [24] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [25] Paul F Kinghorn, Beren Millidge, and Christopher L Buckley. Preventing deterioration of classification accuracy in predictive coding networks. In *International Workshop on Active Inference*, pages 1–15. Springer, 2022.
- [26] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.
- [27] Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. University of London, University College London (United Kingdom), 2003.
- [28] Peter Zeidman, Karl Friston, and Thomas Parr. A primer on variational laplace (vl). *NeuroImage*, 279:120310, 2023.
- [29] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the laplace approximation. *Neuroimage*, 34(1):220–234, 2007.
- [30] Alexander Ororbia and Daniel Kifer. The neural coding framework for learning generative models. *Nature communications*, 13(1):2064, 2022.
- [31] Gaspard Oliviers, Rafal Bogacz, and Alexander Meulemans. Learning probability distributions of sensory inputs with monte carlo predictive coding. *bioRxiv*, pages 2024–02, 2024.
- [32] Umais Zahid, Qinghai Guo, and Zafeirios Fountas. Sample as you infer: Predictive coding with langevin dynamics. *arXiv preprint arXiv:2311.13664*, 2023.
- [33] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [34] Luca Pinchetti, Chang Qi, Oleh Lokshyn, Gaspard Oliviers, Cornelius Emde, Mufeng Tang, Amine M'Charak, Simon Frieder, Bayar Menzat, Rafal Bogacz, et al. Benchmarking predictive coding networks—made simple. *arXiv preprint arXiv:2407.01163*, 2024.

## A Distributions

**Matrix Normal Distribution** The Matrix Normal distribution is a generalization of the multivariate normal distribution to matrix-valued random variables. A random matrix  $\mathbf{W} \in \mathbb{R}^{d_y \times d_x}$  is said to follow a Matrix Normal distribution with mean  $\mathbf{M} \in \mathbb{R}^{d_y \times d_x}$ , row precision  $\Sigma^{-1} \in \mathbb{R}^{d_y \times d_y}$ , and column covariance  $\mathbf{V} \in \mathbb{R}^{d_x \times d_x}$ , denoted as  $\mathcal{MN}(\mathbf{W} \mid \mathbf{M}, \Sigma^{-1}, \mathbf{V})$ , if its probability density function is given by:

$$p(\mathbf{W} \mid \mathbf{M}, \Sigma^{-1}, \mathbf{V}) = \frac{\exp\left(-\frac{1}{2}\text{Tr}(\Sigma^{-1}(\mathbf{W} - \mathbf{M})\mathbf{V}^{-1}(\mathbf{W} - \mathbf{M})^\top)\right)}{(2\pi)^{\frac{d_y d_x}{2}} |\Sigma|^{\frac{d_y}{2}} |\mathbf{V}|^{\frac{d_x}{2}}}. \quad (12)$$

**Wishart Distribution** The Wishart distribution is a distribution over symmetric positive definite matrices and is the conjugate prior for the covariance matrix in multivariate Gaussian distributions. In our formulation, let  $\Sigma^{-1}$  be a random matrix that follows a Wishart distribution with scale matrix  $\Psi$  and degrees of freedom  $\nu$ , denoted as  $\mathcal{W}(\Sigma^{-1} \mid \Psi, \nu)$ . The density function is given by:

$$p(\Sigma^{-1} \mid \Psi, \nu) = \frac{|\Sigma^{-1}|^{\frac{\nu-d_y-1}{2}} \exp\left(-\frac{1}{2}\text{Tr}(\Sigma^{-1}\Psi^{-1})\right)}{2^{\frac{\nu d_y}{2}} |\Psi|^{\frac{\nu}{2}} \Gamma_{d_y}\left(\frac{\nu}{2}\right)}, \quad (13)$$

where  $\Gamma_{d_y}(\cdot)$  denotes the multivariate gamma function.

**Matrix Normal Wishart Distribution** The Matrix Normal Wishart distribution is the joint distribution of  $\mathbf{W}$  and  $\Sigma$ . If  $\mathbf{W} \mid \Sigma^{-1} \sim \mathcal{MN}(\mathbf{W} \mid \mathbf{M}, \Sigma^{-1}, \mathbf{V})$  and  $\Sigma^{-1} \sim \mathcal{W}(\Sigma^{-1} \mid \Psi, \nu)$ , then the joint density is given by:

$$\begin{aligned} \log p(\mathbf{W}, \Sigma \mid \mathbf{M}, \mathbf{V}, \Psi, \nu) &= -\frac{1}{2}\text{Tr}(\Sigma^{-1}(\mathbf{W} - \mathbf{M})\mathbf{V}^{-1}(\mathbf{W} - \mathbf{M})^\top) + \frac{d_x}{2} \log |\Sigma^{-1}| \\ &\quad - \frac{1}{2}\text{Tr}(\Sigma^{-1}\Psi^{-1}) + \frac{\nu - d_y - 1}{2} \log |\Sigma^{-1}| - \frac{d_y d_x}{2} \log(2\pi) \\ &\quad + \frac{d_y}{2} \log |\mathbf{V}^{-1}| - \frac{\nu d_y}{2} \log(2) - \log \Gamma_{d_y}\left(\frac{\nu}{2}\right). \end{aligned} \quad (14)$$

Here,  $\mathbf{M}$  is the mean matrix,  $\mathbf{V}$  is the column covariance matrix,  $\Psi$  is the scale matrix, and  $\nu$  is the degrees of freedom.

## B Derivatives

The derivatives of the energy function from Equation (6) are given by:

$$\nabla_{\mathbf{z}_l} \mathcal{E}_l = \langle \Sigma_l^{-1}(\mathbf{z}_l - \mathbf{W}_l f(\mathbf{z}_{l-1})) \rangle_{q_\lambda(\mathbf{W}_l, \Sigma_l)}, \quad (15)$$

$$\nabla_{\mathbf{z}_l} \mathcal{E}_{l+1} = -\mathbf{D}(\mathbf{z}_l) \langle \mathbf{W}_{l+1}^T \Sigma_{l+1}^{-1} (\mathbf{z}_{l+1} - \mathbf{W}_{l+1} f(\mathbf{z}_l)) \rangle_{q_\lambda(\mathbf{W}_{l+1}, \Sigma_{l+1})}, \quad (16)$$

where  $\mathbf{D}(\mathbf{z}) = \text{diag}(f'(\mathbf{z}))$  is the diagonal Jacobian of the pointwise non-linear transfer function. Under the assumption that  $q_\lambda(\mathbf{W}_l, \Sigma_l)$  is Matrix Normal Wishart, the relevant expectations are given by:

$$\langle \Sigma_l^{-1} \mathbf{W}_l \rangle_{q_\lambda(\mathbf{W}_l, \Sigma_l)} = \nu_l \Psi_l \mathbf{M}_l, \quad (17)$$

$$\langle \mathbf{W}_l^T \Sigma_l^{-1} \mathbf{W}_l \rangle_{q_\lambda(\mathbf{W}_l, \Sigma_l)} = \mathbf{M}_l \nu_l \Psi_l \mathbf{M}_l + d_l \mathbf{V}_l. \quad (18)$$

or equivalently:

$$\nabla_{\mathbf{z}_l} \mathcal{E}_{l+1} = -\mathbf{D}(\mathbf{z}_l) \mathbf{M}_{l+1}^T \langle \Sigma_{l+1}^{-1} \rangle (\mathbf{z}_{l+1} - \mathbf{M}_{l+1} f(\mathbf{z}_l)) \quad (19)$$

$$+ d_{l+1} \mathbf{D}(\mathbf{z}_l) \mathbf{V}_{l+1} f(\mathbf{z}_l), \quad (20)$$

where  $d_l$  is the dimension of  $\mathbf{z}_l$ . The storage and compute costs associated with this approximate posterior can be prohibitively expensive, so it may be wise to use structured approximations to  $q_\lambda(\mathbf{W}_l, \Sigma_l)$  consistent with the quadratic form of  $\mathcal{E}(\mathbf{Z}, \lambda)$ .

## C Predictive Coding

To help clarify the differences between predictive coding (PC) and Bayesian predictive coding (BPC), this section describes PC using the same notation used in Section 2.

PC is a method for inverting hierarchical Gaussian generative models with  $L$  layers of variables  $\mathbf{Z} = \{\mathbf{z}_l : l = 0 \dots L\}$  and parameters  $\Theta = \{\mathbf{W}_l, \Sigma_l : l = 1 \dots L\}$ :

$$\begin{aligned} p(\mathbf{Z} | \Theta) &= p(\mathbf{z}_0) \prod_{l=1}^L p(\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{W}_l, \Sigma_l), \\ p(\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{W}_l, \Sigma_l) &= \mathcal{N}(\mathbf{z}_l | \mathbf{W}_l f(\mathbf{z}_{l-1}), \Sigma_l) \end{aligned} \quad (21)$$

In comparison with Equation 1, Equation 21 does not include priors over parameters  $\Theta$ .

In the PC framework, latent variables  $\mathbf{Z}$  and parameters  $\Theta$  are not given a Bayesian treatment but are instead represented via maximum a posteriori (MAP) and maximum likelihood (ML) estimates, respectively. The estimates are generated via gradient descent, akin to inference in an energy-based model (EBM) where the energy is given by the negative log probability of the model:

$$\begin{aligned} \mathcal{E}(\mathbf{Z}, \Theta) &= -\log \left( p(\mathbf{z}_0) \prod_{l=1}^L p(\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{W}_l, \Sigma_l) \right) \\ &= \frac{1}{2} \sum_{l=1}^L \underbrace{(\mathbf{z}_l - \mathbf{W}_l f(\mathbf{z}_{l-1})) \cdot \Sigma_l^{-1} (\mathbf{z}_l - \mathbf{W}_l f(\mathbf{z}_{l-1}))}_{\mathcal{E}_l} + C \end{aligned} \quad (22)$$

where  $C$  is independent of the  $\mathbf{Z}$ 's and  $\mathcal{E}_l$  is the precision-weighted prediction error. In comparison to Equation 3, Equation 22 is a *function* of the parameters  $\Theta$ , rather than a function of the posterior parameters  $\lambda$ . Moreover, the energy function in Equation 22 does not have an expectation under  $q_\lambda(\Theta)$ . As a result, the resulting expression as precision-weighted prediction errors is identical to Equation 5, modulo the expectation under  $q_\lambda(\Theta)$ .

Akin to BPC, PC is also solved using the expectation-maximisation (EM) algorithm, but now directly on the maximum-likelihood estimates of  $\Theta$  rather than the variational parameters  $\lambda$ :

$$\mathbf{Z}^* = \arg \min_{\mathbf{Z}} \mathcal{E}(\mathbf{Z}, \Theta), \quad \Theta^* = \arg \min_{\Theta} \mathcal{E}(\mathbf{Z}^*, \Theta). \quad (23)$$

To solve these optimisation problems, PC relies on gradient descent for both the latent variables and parameters, where the gradients are given by:

$$\nabla_{\mathbf{z}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{z}_l} = \frac{1}{2} \left( \frac{\partial \mathcal{E}_l}{\partial \mathbf{z}_l} + \frac{\partial \mathcal{E}_{l+1}}{\partial \mathbf{z}_l} \right), \quad \nabla_{\theta_l} = \frac{\partial \mathcal{E}}{\partial \theta_l} = \frac{1}{2} \frac{\partial \mathcal{E}_l}{\partial \theta_l}. \quad (24)$$

The resulting update rule for the  $\mathbf{Z}$  takes the form:

$$\mathbf{z}_l \rightarrow \mathbf{z}_l - \alpha (\Sigma_l^{-1} \mathbf{z}_l - \mathbf{D}(\mathbf{z}_l) \mathbf{W}_{l+1}^T \Sigma_{l+1}^{-1} (\mathbf{z}_{l+1} - \mathbf{W}_{l+1} f(\mathbf{z}_l))) \quad (25)$$

We note that the derivative of the non-linear activation function is bounded by 1 this is a weakly non-linear dynamical system with a block triangular structure. Thus the dynamics dominated by the spectrum of  $\mathbf{A}_l = \Sigma_l^{-1} + \mathbf{W}_{l+1}^T \Sigma_{l+1}^{-1} \mathbf{W}_{l+1}$  allowing us to identify an upper bound on the maximum learning rate parameter as approximately given by the inverse of the maximum eigenvalue of the  $\mathbf{A}_l$  or the sum of the maximum eigenvalues associated with each component, which can be dynamically updated with updates to the posterior distribution over the parameters.

## D Conjugate updates

Here we show that the parameters of  $q(\mathbf{W}_l, \Sigma_l)$  that minimize the energy defined in Equation (3) can be written as a sum of the sufficient statistics of the likelihood  $p(\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{W}_l, \Sigma_l)$  and the natural parameters of the prior  $p(\mathbf{W}_l, \Sigma_l)$ .

Consider the energy function for the  $l^{\text{th}}$  layer of the network, which depends on the  $l$  and  $l - 1^{\text{th}}$  latent variables and the Matrix Normal Wishart parameters of that layer:

$$\mathcal{E}(\mathbf{z}_l, \boldsymbol{\lambda}_l, \mathbf{z}_{l-1}) = \left\langle \log q_{\boldsymbol{\lambda}}(\mathbf{W}_l, \Sigma_l) - \log p(\mathbf{z}_l, \mathbf{W}_l, \Sigma_l | \mathbf{z}_{l-1}) \right\rangle_{q_{\boldsymbol{\lambda}}(\mathbf{W}_l, \Sigma_l)} \quad (26)$$

We obtain the optimal value of the parameters, which we denote  $\boldsymbol{\lambda}_l^*$ , by setting the derivative of this local energy function to 0 and solving for  $\boldsymbol{\lambda}_l$ . The conjugacy of the likelihood and prior  $p(\mathbf{z}_l, \mathbf{W}_l, \Sigma_l | \mathbf{z}_{l-1})$ , combined with the Matrix Normal Wishart form of the variational posteriors over  $\mathbf{W}_l, \Sigma_l$ , means that the variational parameters  $\boldsymbol{\lambda}_l$  can be equated with the natural parameters of a Matrix Normal Wishart distribution, i.e.,

$$\boldsymbol{\lambda}_l \equiv \boldsymbol{\eta}_l$$

$$\boldsymbol{\eta}_l = \begin{bmatrix} \mathbf{V}_l^{-1} \\ \mathbf{M}_l \mathbf{V}_l^{-1} \\ \boldsymbol{\Phi}_l + \mathbf{M}_l \mathbf{V}_l^{-1} \mathbf{M}_l^\top \\ \nu_l - d_y + d_x - 1 \end{bmatrix} \quad (27)$$

The value of  $\boldsymbol{\eta}_l$  at which the derivative of Equation (26) vanishes, results in a succinct expression in terms of the natural parameters of the prior  $\boldsymbol{\eta}_0$  and the sufficient statistics of the  $\mathbf{z}_{l-1}$ -conditioned likelihood over  $\mathbf{z}_l$ :

$$\boldsymbol{\eta}_l^* = \begin{bmatrix} \mathbf{V}_{l,0}^{-1} + f(\mathbf{z}_{l-1})f(\mathbf{z}_{l-1})^\top \\ \mathbf{M}_{l,0} \mathbf{V}_{l,0}^{-1} + f(\mathbf{z}_{l-1})\mathbf{z}_l^\top \\ \boldsymbol{\Phi}_{l,0} + \mathbf{M}_{l,0} \mathbf{V}_{l,0}^{-1} \mathbf{M}_{l,0}^\top + \mathbf{z}_l \mathbf{z}_l^\top \\ \nu_{l,0} - d_y + d_x - 1 + 1 \end{bmatrix} \quad (28)$$

The  $\mathbf{X}_{l,0}$  notation denotes parameters of the Matrix Normal Wishart prior for the  $l^{\text{th}}$  layer. Because this model is fully conjugate, this update is an exact Bayesian update and thus the optimal variational posterior is equal to the true posterior [27].

## E Related work

Several extensions to predictive coding (PC) have previously been proposed. Variational Laplace [28, 29] employs a Laplace approximation, resulting in an algorithm expressed in terms of precision-weighted prediction errors. This approach additionally includes priors on model parameters, resulting in maximum *a posteriori* (MAP) estimates. However, it does not represent full posterior distributions over parameters, thus limiting its ability to quantify epistemic uncertainty. Other approaches have explored incorporating sparse priors into the generative model [30], but these also do not infer full posterior distributions. Variants of PC employing Monte Carlo sampling and Langevin dynamics [31, 32] have successfully inferred posterior distributions over latent variables, but have not been extended to infer distributions over parameters, again limiting their capacity to capture epistemic

uncertainty. An interesting direction for future research would be to integrate our Bayesian predictive coding (BPC) algorithm with such sampling-based approaches, potentially leading to a fully Bayesian PC method that infers posterior distributions over both latent variables and parameters.

## F Experiment details

### E.1 Accuracy

In this section, we provide additional details regarding the experiments presented in Section 3.1. For all datasets, reported performance is averaged over 5 random seeds. In the regression task (energy), we report performance using the mean squared error (MSE) between the predicted and true values. For classification tasks (MNIST and two moons), we use one-hot encoded labels during training. At test time, the predicted class label is determined by taking the argmax over the output nodes, where the number of nodes corresponds to the number of classes.

For the energy and MNIST datasets, we employ the same neural network architecture. Specifically, we use a four-layer neural network with 128 hidden units per layer and ReLU activations. Training is performed using mini-batches of size 128. We initialized the linear weights  $\mathbf{W}$  of shape (out\_features, in\_features) from a uniform distribution  $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ , where  $k = \frac{1}{\text{in\_features}}$ . Similarly, the bias  $\mathbf{b}$  of shape (out\_features) is initialized from  $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ .

For BP, we used the Adam [24] optimizer with a learning rate of 0.001. For PC, we used the AdamW [33] optimizer for the parameters with a learning rate of 0.0002 and a weight decay value of 0.65, and a stochastic gradient descent optimiser for the hidden states with a learning rate of 0.01 and momentum of 0.65. These parameter values followed those proposed in [34] for the same dataset. We used 10 iterations of hidden state updates for each batch, before performing one gradient step on the weights.

For BPC, we used the Adam [24] optimizer for hidden states, with a learning rate of 0.01 and 10 iterations per batch. For the parameter learning rate, we used  $\kappa_t = t^{-\epsilon}$ , where  $t$  is the total number of updates and  $-\epsilon$  was set to 0.25.

We set the prior over the weights  $\mathbf{M}^{(0)}$  to be a matrix of zeros of the appropriate size. We set the prior over  $\mathbf{V}^{(0)}$  to be  $10 \cdot \mathbf{I}$  where  $\mathbf{I}$  is an identity matrix of the appropriate size and set  $\Psi^{(0)}$  to be  $1000 \cdot \mathbf{I}$ . Finally, we set  $\nu^{(0)}$  to be  $d_y + 2$ . All initial estimates of the posterior natural parameters  $\eta$  were set to the same as the prior, besides  $\mathbf{M}$  which uses the initialisation described for  $\mathbf{W}$ .

The hyper-parameters for the two moons dataset experiments remain the same, except that we use a smaller network architecture consisting of a single hidden layer with 100 hidden units.

### E.2 Synthetic regression

In the synthetic regression tasks, we use the same parameter settings used in two moons. For the aleatoric uncertainty plot, we generate samples from:

$$\mathbf{y} = -(\mathbf{x} + 0.5) \cdot \sin(3\pi\mathbf{x}) + \mathcal{N}(0, (0.45 \cdot (\mathbf{x} + 0.5))^2),$$

where  $\mathbf{x}$  is normally distributed around zero. For the epistemic uncertainty plot, we use:

$$\mathbf{y} = \mathbf{x}^3 + \mathcal{N}(0, 9),$$

where  $\mathbf{x}$  is uniformly sampled with half of the values drawn from the interval [3, 5] and the other half from [-5, -3].

### E.3 UCI datasets

For the UCI datasets, we use the same parameter settings as in previous experiments, but with networks that have two hidden layers and 50 hidden nodes. For Bayes by Backprop (BBB) [26], we use the Adam [24] optimizer with a learning rate of 0.001, a prior mean of 0.0, a  $\sigma$  of 1.0, and a batch

size of 100. When computing the log predictive density (LPD) for both BBB and BPC, we draw 20 posterior samples per data batch.