On permutation symmetries in Bayesian neural network posteriors: Appendix

Anonymous Author(s)
Affiliation
Address
email

A Additional details on the alignment method

In the main paper, to align the distributions with respect to permutation matrices we argue to use the Wasserstein distance rather than the Kullback-Leibler (KL) divergence. Indeed, by considering the KL divergence $\mathrm{KL}\left[P_{\#}q_1 \parallel q_0\right]$ between Gaussians we have

$$\mathrm{KL}\left[P_{\#}q_1 \parallel q_0\right] = \log\det \mathrm{diag}(\boldsymbol{s}_0) - \log\det \mathrm{diag}(\boldsymbol{P}\boldsymbol{s}_1) + \mathrm{Tr}\big(\mathrm{diag}(\boldsymbol{P}\boldsymbol{s}_1\boldsymbol{s}_0^{-1})\big) + \tag{1}$$

$$(\boldsymbol{m}_0 - \boldsymbol{P}\boldsymbol{m}_1)^\top \mathrm{diag}(\boldsymbol{s}_0^{-1})(\boldsymbol{m}_0 - \boldsymbol{P}\boldsymbol{m}_1) \tag{2}$$

It's easy to verify that the first three terms do not depend on $\boldsymbol{P}$, leading to just a distance between means and disregarding any covariance information. In the figure below, we visualize the difference between doing linear assignment problem (LAP) with the KL cost and LAP with the Wasserstein cost.

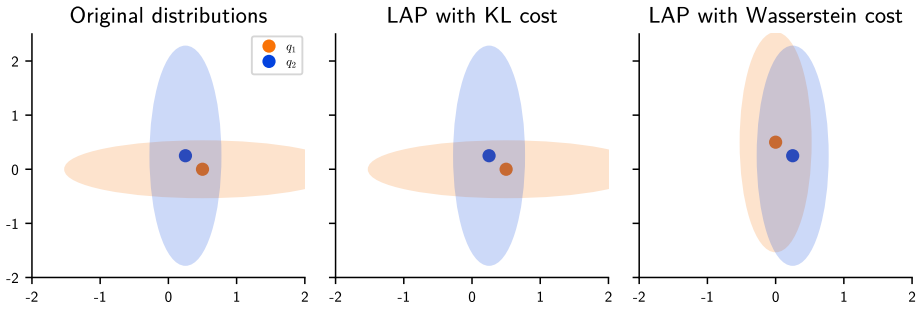

Figure A1: **Alignment using different objectives.** Given two distributions symmetrical w.r.t. the $y = x$ plane, using the KL cost LAP results in the identity permutation (which fails to recover the symmetry), while the Wasserstein cost better aligns the two distributions

B Experimental setup

Tables 2 and 3 show details on the multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) base architectures used in our experimental campaign, while Table 1 reports the hyper-parameters used in the experiments. Note that differently from Entezari et al. [5] and Ainsworth et al. [2], we don't use data augmentation. A possible protocol for handling data augmentation in Bayesian neural networks (BNNs) is presented by Osawa et al. [14] and involves carefully tuning the likelihood temperature to correctly counting the number of data points.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Table 1: Hyperparameters used for the experiments

Dataset	CIFAR10		MNIST
Model	ResNet20	MLP	MLP
Data Aug.	False	False	False
Batch size	500	500	500
Temperature	1.0	1.0	1.0
Test samples	128	128	128
Train samples	1	1	1
VI std. init	0.01	0.01	0.01
Base features	16	512	512
Prior var	0.01	0.0025	0.01
Learning rate	0.000001	0.000001	0.000001
Train epochs	1000	1000	1000

Table 2: MLP

Layer	**Dimensions**
Linear-ReLU	$512 \times D_{\text{in}}$
Linear-ReLU	512×512
Linear-ReLU	512×512
Linear-Softmax	$D_{\text{out}} \times 512$

Table 3: ResNet20

Layer	**Dimensions**
Conv2D	$16 \times 3 \times 3 \times D_{\text{in}}$
Residual Block	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 3$
Residual Block	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$
Residual Block	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
AvgPool	8×8
Linear-Softmax	$D_{\text{out}} \times 64$

B.1 Computing platform

The experiments have been performed using JAX [4] and run on two AWS p4d.24xlarge instances with 8 NVIDIA A100 GPUs. Experiments were conducted using in the eu-west-1 region, which has a carbon efficiency of 0.62 kgCO_2eq/kWh. A cumulative of 6500 hours of computation was performed on GPUs and it includes interactive sessions as well as small experiments with very low GPU usage, providing a pessimistic estimation of the true utilization. Total emissions are estimated to be 1007.5 kgCO_2eq of which 100 percents were directly offset by AWS.

C Additional results

We present timings obtained by profiling the time needed to solve the sum of bilinear assignment problems (SOLAP) with the Wasserstein cost, as well as the time for the deterministic case [2]. In Fig. A2 we show the results for MLP and ResNet20 architectures, varying the model width. It is evident that, in the majority of cases, the algorithm completes within a minute. Moreover, as anticipated, in case of variational inference (VI) solving our distribution alignment problem for wide neural networks is more computationally demanding compared to merely matching weights from stochastic gradient descent (SGD) solutions.

Finally, we also test our setup on the CIFAR100 dataset [11]. Surprisingly, we were not able to replicate the same level of performance as in the other cases. In Fig. A3, we see that, despite converging well, we fall short to find zero-barrier solutions. Similarly to the comments of Ainsworth et al. [2], we also stress that the failure to align distributions does not rule out the existence of a proper permutation map that the algorithm couldn't find. Nonetheless, this raises a number of questions: the Bayesian posterior is the product of two ingredients, the prior and the likelihood, conditioned to observing a dataset.

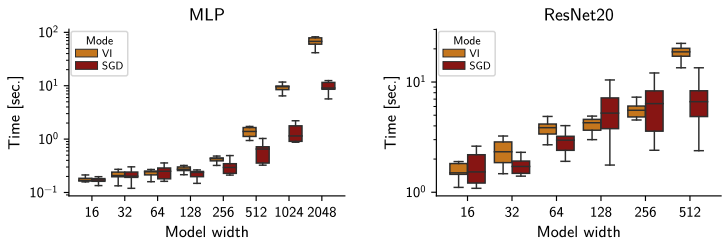

Figure A2: **Timings.** Profile of the algorithms to align the VI solutions and to match weights from SGD solutions.

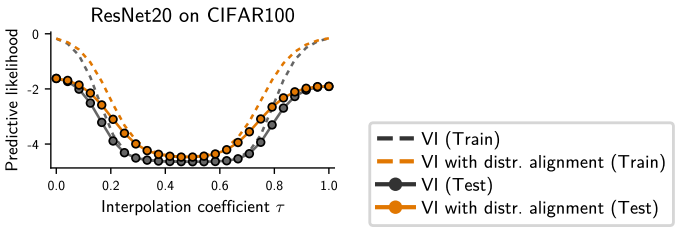

Figure A3: **Alignment failure.** The method proposed fails to recover zero-barrier solutions for CIFAR100.

D A primer on variational inference for Bayesian neural networks

VI is a classic tool to tackle intractable Bayesian inference [9, 3]. VI casts the inference problem into an optimization-based procedure to compute a tractable approximation of the true posterior. Assume a generic parametric model f parameterized by some unknown parameters $\boldsymbol{\theta}$ (i.e. $f(\cdot, \boldsymbol{\theta})$) and a collection of data $\boldsymbol{y} \in \mathbb{R}^N$ corresponding to some input points $\boldsymbol{X} = \{\boldsymbol{x}_i \,|\, \boldsymbol{x}_i \in \mathbb{R}^{D_{\text{in}}}\}_{i=1,\dots,N}$. In our setting, we have a probabilistic model $p(\boldsymbol{y} \,|\, f(\boldsymbol{X}; \boldsymbol{\theta}))$ with parameters $\boldsymbol{\theta}$, a prior distributions on them $p(\boldsymbol{\theta})$ and a set of observations $\{\boldsymbol{X}, \boldsymbol{y}\}$. In a nutshell, the general recipe of VI consists of (i) introducing a set $\mathcal{Q}$ of distributions; (ii) defining a tractable objective that "measure" the distance between any arbitrary distribution $q(\boldsymbol{\theta}) \in \mathcal{Q}$ and the true posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{y})$; and finally (iii) providing a programmatic way to find the distribution $\widetilde{q}(\boldsymbol{\theta})$ that minimizes such distance. In practice, $q(\boldsymbol{\theta})$ has some free parameters $\boldsymbol{\nu}$ (also known as *variational parameters*), which are optimized such that the approximating distribution $q(\boldsymbol{\theta}; \boldsymbol{\nu})$ is as closer as possible to the true posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{y})$. We can derive the variational objective starting from the definition of the KL,

$$\begin{aligned} \text{KL}\left[q(\boldsymbol{\theta}; \boldsymbol{\nu}) \parallel p(\boldsymbol{\theta} \,|\, \boldsymbol{y})\right] &= \mathbb{E}_{q(\boldsymbol{\theta}; \boldsymbol{\nu})}\left[\log q(\boldsymbol{\theta}; \boldsymbol{\nu}) - \log p(\boldsymbol{\theta} \,|\, \boldsymbol{y})\right] = \\ &= \mathbb{E}_{q(\boldsymbol{\theta}; \boldsymbol{\nu})}\left[\log q(\boldsymbol{\theta}; \boldsymbol{\nu}) - \log p(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right] + \log p(\boldsymbol{y}) \end{aligned} \tag{3}$$

Rearranging we have that

$$\log p(\boldsymbol{y}) - \text{KL}\left[q(\boldsymbol{\theta}; \boldsymbol{\nu}) \parallel p(\boldsymbol{\theta} \,|\, \boldsymbol{y})\right] = \mathbb{E}_{q(\boldsymbol{\theta}; \boldsymbol{\nu})}\left[\log q(\boldsymbol{\theta}; \boldsymbol{\nu}) - \log p(\boldsymbol{y} \,|\, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right] \tag{4}$$

The r.h.s. of the equation defines our variational objective, also known as evidence lower bound (ELBO), that can be arranged as follows,

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\nu}) = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}; \boldsymbol{\nu})} \log p(\boldsymbol{y} \,|\, \boldsymbol{\theta})}_{\text{Model fitting term}} - \underbrace{\text{KL}\left[q(\boldsymbol{\theta}; \boldsymbol{\nu}) \parallel p(\boldsymbol{\theta})\right]}_{\text{Regularization term}} . \tag{5}$$

This formulation highlights the property of this objective, which is made of two components: the first one is the expected log-likelihood under the approximate posterior q and measures how the model fits the data. The second term, on the other hand, has the regularization effect of penalizing posteriors that are far from the prior as measured by the KL. Before diving into the challenges of optimization of the ELBO, we shall spend a brief moment discussing the form of the approximating distribution q. One of the simplest and easier choice is the mean field approximation [7], where each variable θ_i is taken to be independent with respect to the remaining $\boldsymbol{\theta}_{-i}$. Effectively, this imposes a factorization of the posterior,

$$q(\boldsymbol{\theta}; \boldsymbol{\nu}) = \prod_{i=1}^{K} q(\theta_i; \boldsymbol{\nu}_i) \tag{6}$$

where $\boldsymbol{\nu}_i$ is the set of variational parameters for the parameter θ_i. On top of this approximation, $q(\theta_i)$ is often chosen to be Gaussian,

$$q(\theta_i) = \mathcal{N}(\mu_i, \sigma_i^2) \tag{7}$$

Now, the collection of all means and variances $\{\mu_i, \sigma_i^2\}_{i=1}^K$ defines the set of variational parameters to optimize.

For BNNs the analytic evaluation of the ELBO (and its gradients) is always untractable due the non-linear nature of the expectation of the log-likelihood under the variational distribution. Nonetheless, this can be easily estimated via Monte Carlo integration [13], by sampling N_{MC} times from $q_{\boldsymbol{\nu}}$,

$$\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\nu})} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) \approx \frac{1}{N_{\text{MC}}} \sum_{j=1}^{N_{\text{MC}}} \log p(\boldsymbol{y} \mid \widetilde{\boldsymbol{\theta}}_j) \,, \quad \text{with} \quad \widetilde{\boldsymbol{\theta}}_j \sim q(\boldsymbol{\theta}; \boldsymbol{\nu}) \tag{8}$$

In practice, this is as simple as re-sampling the weights and the biases for all the layers N_{MC} times and computing the output for each new sample.

We now have a tractable objective that needs to be optimized with respect to the variational parameters $\boldsymbol{\nu}$. Very often the KL term is known, making its differentation trivial. On the other hand the expectation of the likelihood is not available, making the computation of its gradients more challenging. This problem can be solved using the so-called *reparameterization trick* [19, 10]. The reparameterization trick aims at constructing $\boldsymbol{\theta}$ as an invertible function $\mathcal{T}$ of the variational parameters $\boldsymbol{\nu}$ and of another random variable $\boldsymbol{\varepsilon}$, so that $\boldsymbol{\theta} = \mathcal{T}(\boldsymbol{\varepsilon}; \boldsymbol{\nu})$. Generally, a $\mathcal{T}$ that suits this constraint might not exists; Ruiz et al. [18] discuss how to build "weakly" dependent transformation $\mathcal{T}$ for distributions like Gamma, Beta and Log-normal. For discrete distributions, instead, one could use a continuous relaxation, like the Concrete [12]. $\boldsymbol{\varepsilon}$ is chosen such that its marginal $p(\boldsymbol{\varepsilon})$ does not depend on the variational parameters. With this parameterization, $\mathcal{T}$ separates the deterministic components of q from the stochastic ones, making the computation of its gradient straightforward. For a Gaussian distribution with mean μ and variance σ^2, $\mathcal{T}$ corresponds to as simple scale-location transformation of an isotropic Gaussian noise,

$$\theta \sim \mathcal{N}(\mu, \sigma^2) \iff \theta = \mu + \sigma\varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, 1) \,. \tag{9}$$

This simple transformation ensures that $p(\varepsilon) = \mathcal{N}(0, 1)$ does not depends on the variational parameters $\boldsymbol{\nu} = \{\mu, \sigma^2\}$. The gradients of the ELBO can be therefore computed as

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) \mid_{\boldsymbol{\theta} = \mathcal{T}(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \mathcal{T}(\boldsymbol{\varepsilon}; \boldsymbol{\nu}) \right] - \nabla_{\boldsymbol{\nu}} \text{KL} \left[q(\boldsymbol{\theta}; \boldsymbol{\nu}) \parallel p(\boldsymbol{\theta}) \right] \,. \tag{10}$$

The gradient $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{\theta})$ depends on the model and it can be derived with automatic differentation tools [1, 15], while $\nabla_{\boldsymbol{\nu}} \mathcal{T}(\boldsymbol{\varepsilon}; \boldsymbol{\nu})$ doesn't have any stochastic components and therefore can be known deterministically. Note that the reparameterization trick can be also used when the KL is not analitically available. In that case, we would end up with,

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p(\boldsymbol{\varepsilon})} \left[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) + \log q(\boldsymbol{\theta}; \boldsymbol{\nu}) - \log p(\boldsymbol{\theta}) \right]_{\boldsymbol{\theta} = \mathcal{T}(\boldsymbol{\varepsilon};\boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \mathcal{T}(\boldsymbol{\varepsilon}; \boldsymbol{\nu}) \tag{11}$$

Roeder et al. [17] argue that when we believe that $q(\boldsymbol{\theta}; \boldsymbol{\nu}) \approx p(\boldsymbol{y} \mid \boldsymbol{\theta})$, Eq. (11) should be prefered over Eq. (10) even if computing analitically the KL is possible. Note that this case is very unlikely for BNN posteriors, and that the additional randomness introduced by the Monte Carlo estimation of the KL could be harmful.

In case of large datasets and complex models, the formulation summarized in Eq. (10) can be computationally challenging, due to the evaluation of the likelihood and its gradients N_{MC} times. Assuming factorization of the likelihood,

$$p(\boldsymbol{y} \mid \boldsymbol{\theta}) = p(\boldsymbol{y} \mid f(\boldsymbol{X}; \boldsymbol{\theta})) = \prod_{i=1}^{N} p(y_i \mid f(\boldsymbol{x}_i; \boldsymbol{\theta})) \tag{12}$$

this quantity can be approximated using mini-batching [6, 8]. Recalling $\boldsymbol{y}$ as the set of labels of our dataset with N examples, by taking $\mathcal{B} \subset \boldsymbol{y}$ as a random subset of $\boldsymbol{y}$, the likelihood term can be estimated in an unbiased way as

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{\theta}) \approx \frac{N}{M} \sum_{y_i \sim \mathcal{B}} \log p(y_i \mid \boldsymbol{\theta}) \,. \tag{13}$$

where M is the number of points in the minibatch. At the cost of increase "randomness", we can use Eq. (10) to compute the gradients of the ELBO with the minibatch formulation in Eq. (13). Stochastic optimization, e.g. any version of SGD, will converge to a local optimum provided with a decreasing learning rate and sufficient gradient updates [16].

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[2] S. Ainsworth, J. Hayase, and S. Srinivasa. Git Re-Basin: Merging Models modulo Permutation Symmetries. In *The Eleventh International Conference on Learning Representations*, 2023.

[3] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A Review for Statisticians. 112 (518):859–877, 2017.

[4] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: Composable Transformations of Python+NumPy Programs, 2018.

[5] R. Entezari, H. Sedghi, O. Saukh, and B. Neyshabur. The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks. In *International Conference on Learning Representations*, 2022.

[6] A. Graves. Practical Variational Inference for Neural Networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

[7] G. E. Hinton and D. van Camp. Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, 1993. ISBN 0897916115.

[8] M. D. Hoffman, D. M. Blei, C. Wang, and J. W. Paisley. Stochastic Variational Inference. 14(1):1303–1347, 2013.

[9] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. 37(2):183–233, 1999-11-01.

[10] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

[11] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[12] C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net, 2017.

[13] N. Metropolis and S. Ulam. The monte carlo method. 44(247):335–341, 1949. PMID: 18139350.

[14] K. Osawa, S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota. Practical Deep Learning with Bayesian Principles. In *Advances in Neural Information Processing Systems*, volume 32, pages 4287–4299. Curran Associates, Inc., 2019.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[16] H. Robbins and S. Monro. A Stochastic Approximation Method. 22(3):400–407, 1951.

[17] G. Roeder, Y. Wu, and D. Duvenaud. Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6925–6934, 2017.

[18] F. R. Ruiz, M. Titsias, and D. Blei. The Generalized Reparameterization Gradient. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[19] T. Salimans and D. A. Knowles. Fixed-Form Variational Posterior Approximation through Stochastic Linear Regression. 8(4):837–882, 2013.