Subspace Inference for Bayesian Deep Learning

Pavel Izmailov [*1] **Wesley Maddox** [*1] **Polina Kirichenko** [*1] **Timur Garipov** [*2] **Dmitry Vetrov** [2 3]
Andrew Gordon Wilson [1]

Abstract

Bayesian inference was once a gold standard for learning with neural networks, providing accurate full predictive distributions and well calibrated uncertainty. However, scaling Bayesian inference techniques to deep neural networks is challenging due to the high dimensionality of the parameter space. In this paper, we construct low-dimensional subspaces of parameter space that contain diverse sets of models, such as the first principal components of the stochastic gradient descent (SGD) trajectory. In these subspaces, we are able to apply elliptical slice sampling and variational inference, which struggle in the full parameter space. We show that Bayesian model averaging over the induced posterior in these subspaces produces high accurate predictions and well-calibrated predictive uncertainty for both regression and image classification.

1. Introduction

Bayesian methods were once the state-of-the-art approach for inference with neural networks (MacKay, 2003; Neal, 1996a). However, the parameter spaces for modern deep neural networks are extremely high dimensional, posing challenges to standard Bayesian inference procedures.

In this paper we propose a different approach to approximate Bayesian inference in deep learning models, termed **D**imensionality **R**educed **Bayes** (DR Bayes). We design a low-dimensional subspace $\mathcal{S}$ of the weight space and approximate the posterior over the parameters of the model within this subspace. It is our contention that this subspace $\mathcal{S}$ can be chosen to contain a diverse set of models, over which Bayesian model averaging leads to accurate predictions and well-calibrated uncertainties.

In Figure 1 we visualize the samples from the approximate posterior and the corresponding predictive distributions constructed by our method for a 10-dimensional random subspace, and a rich 2-dimensional subspace containing a low-loss curve between two independently trained SGD solutions (see Garipov et al., 2018) on a synthetic 1-dimensional regression problem. As we can see, the predictive distribution corresponding to a random subspace does not capture a diverse set of possible trajectories, but by sampling from the posterior in the rich subspace we are able to get meaningful uncertainty over predictions.

Our paper is structured as follows. In Section 2 we describe the proposed method for inference in low-dimensional subspaces of the parameter space. We analyze the effects of using different subspaces and approximate inference methods on a synthetic regression problem in Section 3.1 and on image classification problems (CIFAR-10 and CIFAR-100) in Section 3.2. Overall, the method achieves strong performance in terms of both accuracy and likelihood. In particular, we construct 5-dimensional subspaces that contain enough diversity for approximate Bayesian model averaging on a 36 million dimensional WideResNet trained on CIFAR-100. Additional results on automatic choice of dimensionality of the subspace can be found in the appendix.

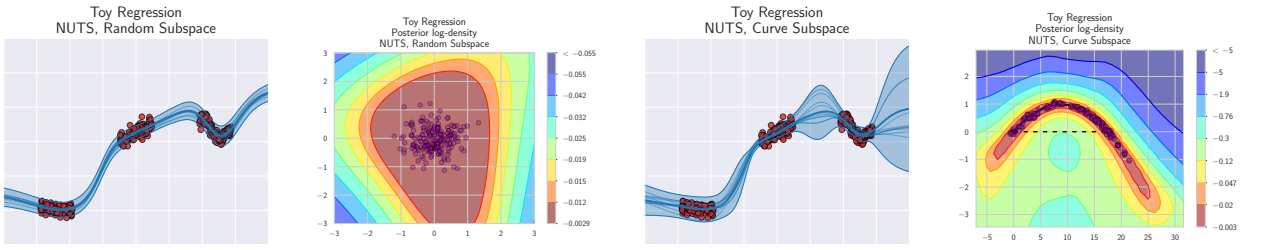
2. Inference within a Subspace

Algorithm 1 Approximate Bayesian inference using a subspace

Input: data, $\mathcal{D}$; model, $\mathcal{M}$;
Train $\mathcal{M}$ on $\mathcal{D}$ with stochastic gradient descent.
Construct subspace, i.e. using Algorithm 2.
Perform inference within subspace using Eq. 3 and prior with approximate Bayesian inference procedure, see Section 2.2.

[*]Equal contribution [1]Cornell University [2]Samsung AI Center in Moscow [3]National Research University Higher School of Economics. Correspondence to: Wesley Maddox <wm326@cornell.edu>.

Presented at the ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning.



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

Figure 1: Predictive distribution and samples in the parameter space for our method on a synthetic regression problem in a random subspace **(a, b)** and subspace containing a constant-loss curve between two independently trained soultions **(c, d)** (see Garipov et al., 2018, for details). On the plots **(a, c)** data points are shown with red circles, the shaded region represents the 3σ-region of the predictive distribution at each point, and the predictive mean is shown with a thick blue line and sample trajectories are shown with thin blue lines. Panels **(b, d)** show the contour plots of the posterior within the corresponding subspace; magenta circles represent samples from the posterior in the subspace. In the rich subspace containing the constant loss curve, the samples produce better uncertainty estimates and more diverse trajectories. We use a small fully-connected network with 4 hidden layers. See Section 3.1 for more details.

2.1. Model Definition

To perform inference within a subspace, we assume a set of $K+1$ vectors: $\{v_1, v_2, \ldots, v_K, \hat{w}\}$ in the full weight space (e.g. $\mathbb{R}^p$); we will do inference in a subspace, $\mathcal{S}$, of the full weight space, defined by the projection matrix P:

$$\begin{aligned}\mathcal{S} &= \{w | w = \hat{w} + t_1 v_1 + \ldots t_K v_K\} \\ &= \{w | w = \hat{w} + Pt\},\end{aligned} \tag{1}$$

where $P = (v_1^T, \ldots, v_K^T) \in \mathbb{R}^{p \times K}$, and $t = (t_1, \ldots, t_K)^T \in \mathbb{R}^K$.

Performing Bayesian inference in the subspace requires defining a new model over $t \in \mathbb{R}^K$; these serve as coefficients for vectors v_i. The new model has the likelihood function:

$$p(\mathcal{D}|t) = p_M(\mathcal{D}|w = \hat{w} + Pt), \tag{2}$$

where the right-hand-side represents the likelihood for the model, $\mathcal{M}$, with parameters $\hat{w} + Pt$ and $\mathcal{D}$ is the dataset.

We can then do approximate Bayesian inference over the parameters, t, after choosing a prior, $p(t)$, on these parameters. As we can set the number K of parameters to be much smaller than the dimensionality p of the full parameter space, performing Bayesian inference becomes considerably more tractable (i.e. only large data issues must be dealt with instead of large data and large dimensionality). The procedure is summarized in Algorithm 1, with construction of subspaces, P, being left to Appendix C and further computational benefits in Appendix B.

2.2. Sampling

Once we have formed an approximate posterior distribution $q \approx p(t|\mathcal{D})$, we can use this distribution to do sampling in the original parameter space. To do so we first sample $\tilde{t} \sim q$, and then construct a sample in the original space as $\tilde{w} = \hat{w} + P\tilde{t}$.

For example, if we were to consider the posterior predictive distribution on new data points, $\mathcal{D}$, then we would be able to compute a Monte Carlo estimate of the integral

$$p(\mathcal{D} \ |\mathcal{D}) = \int p_M(\mathcal{D} \ |\tilde{w} = \hat{w} + Pt) p(t|\mathcal{D}) dt,$$

using samples from $p(t|\mathcal{D})$ or q. Further description of the samplers used are in Appendix F.

2.3. Potential Issues

We next outline two potential failings of performing inference within a subspace. The first is related to the transformation itself, while the second deals with posterior concentration.

Loss of Measure When mapping into a lower-dimensional subspace of the true parameter space, we lose the ability to invert the transform and thus measure (i.e. volume of the distribution) is lost. For a more intuitive explanation, consider the following simple example in $\mathbb{R}^2$. Form a spherical density, $p(x, y) = N(0, I_2)$, and then fix x and y along a slice, that is $x - y = c$. However, the resultant distribution has *no area* (it is a line, just a slice with no width).

For this reason, it is better to not consider the subspace projected model a re-parameterized version of the same model[1], but to consider this an *entirely different model that just happens to share many of the same functional capabilities* as the fully parameterized model.

To use a full rank posterior, Maddox et al. (2019) proposed to use a low rank plus diagonal approximation to the covariance; however, here, we embrace the abyss of lost measure. Alternatively, we could seek to minimize the loss of measure by constructing a minimal distance mapping (perhaps a Wasserstein distance) as in Patra and Dunson (2018).

Posterior Concentration Under some assumptions, for a fixed dimensionality of parameters, p, in the limit of infinite data, that is $N \to \infty$, the well-known Bernstein von Mises theorem from asymptotic statistics (e.g. Chapter 10 of Vaart (1998)) states that the posterior distribution concentrates at the maximum likelihood estimate with covariance given by the inverse of the Fisher information. Due to singularity of the Fisher information for DNNs (and many other machine learning models, e.g. Watanabe (2007)[2]), we would not expect the posterior of fully parameterized DNNs to concentrate in this manner. Rather we would expect it to converge to connected point masses and gorges of global minima, the degenerate extension of the paths explored in Garipov et al. (2018).

By contrast, in the model proposed in Section 2.1, there are only $K << N$ parameters as opposed to $p >> N$ parameters in the full weight space, while the number of observed data points, N is constant. In this under-parametrized setting the posterior concentration becomes an issue, as there is very little diversity in samples from the posterior.

2.4. Preventing Posterior Concentration with Fixed Temperature Posteriors

To address the issue of posterior concentration in the subspace, we propose to introduce temperature for the likelihood term in the posterior. More precisely, we utilize the *tempered* posterior:

$$p_T(t|\mathcal{D}) \propto \underbrace{p(\mathcal{D}|t)^{1/T}}_{\text{likelihood}} \underbrace{p(t)}_{\text{prior}}. \tag{3}$$

When $T = 1$ the true posterior is recovered, and as $T \to \infty$, the tempered posterior p_T would approach the prior $p(t)$.

Tempered posteriors are often used in Bayesian inference algorithms to enhance multi-modal explorations (e.g., Geyer, 1991; Neal, 1996b). Similarly, Watanabe (2013) uses a tempered posterior to recover an expected generalization error of Bayesian models. In the subspace inference setting, we would not necessarily want $T = 1$ because of the possibility of posterior concentration when constructing the subspace. As a heuristic, we propose to set $T = N/K$, where N is the number of observed datapoints. For this setting of T the likelihood term in (3) has the same effect on the posterior as if the number of data points was K, the number of projected parameters.

3. Experiments

In this section, we evaluate the proposed method empirically using different approximate inference methods for sampling from the posterior in the subspace (discussed in Appendix F). We show that approximate Bayesian inference within a subspace gives good predictive uncertainties on a toy regression problem, UCI regression tasks (see Appendix G.1) and large scale image classification on CIFAR-10 and CIFAR-100. Overall, we demonstrate the applicability of the proposed method across a wide range of datasets.

3.1. Toy Regression

In order to gain intuition about the effect of different subspaces and inference methods we first apply the proposed method to a synthetic 1-dimensional regression problem. We use a fully-connected architecture with hidden layers with 200, 50, 50, 50 neurons respectively. The network takes two inputs: x and x^2 and outputs a single real value $y = f(x)$. While this feature representation is redundant, we found that it makes it easier to train the network.

To generate the data we set the weights of the network with this same architecture randomly, and evaluate the predictions $f(x)$ for 400 points sampled uniformly in intervals $[-7.2, -4.8]$, $[-1.2, 1.2]$, $[4.8, 7.2]$. We add Gaussian noise to the outputs $y = f(x) + \epsilon(x)$. The data is visualized with red circles in Figure 3.

We train an SWA-solution (Izmailov et al., 2018), and construct 3 subspaces – a 10-dimensional random subspace, 10-dimensional PCA-subspace and a 2-dimensional curve subspace (see Section C). We then run variational inference, elliptical slice sampling, and MCMC (see Appendix F) in each of the subspaces. We visualize the predictive distributions for each combination of method and subspace in Figure 10 and samples

[1] We cannot construct a Jacobian matrix and take its determinant and apply this to correct the density.

[2] For example, ReLU neural networks are invariant to scaling, which immediately produces singular Hessian matrices and then singular Fisher matrices.

in Figure 11.

In the random and PCA subspaces the shape of the posterior is relatively similar to Gaussian, and all methods are able to produce reasonable samples. In the curve subspace the posterior has a more complex shape, and the variational methods were unable to produce a good fit. Among Monte Carlo methods elliptical slice sampling was able to produce a better fit on the curve subspace. In the subsequent experiments we use simple VI and eliptical slice sampling, as these methods scale better than NUTS and empirically worked well (except for VI on the curve subspace).

In the bottom row of Figure 3 we visualize the predictive distributions for elliptical slice sampling in each of the subspaces. In the random subspace the predictive distribution does not capture a diverse set of models. In particular, the variance of the predictive distributions does not increase outside the regions containing data points. In the PCA subspace on the other hand, the uncertainty representation is much more rich. The variance of the predictive distribution is large far from the data, and small near the data. Finally, the curve subspace contains the most diverse set of models, and correspondingly ESS produces qualitatively an even better predictive distribution in this subspace.

In the top row of Figure 3 we visualize the predictive distributions for simple variational inference applied in the original parameter space (as opposed to a subspace), Gaussian Process with an RBF kernel and SWAG (Maddox et al., 2019) for comparison. SWAG predictive distribution is similar to the predictive distribution of ESS in the PCA subspace, as both methods attempt to approximate the posterior in the subspace containing the principal components of the SGD trajctory. Variational inference in the initial parameter space fails to provide a meaningful uncertainty representation. A Gaussian process with a RBF kernel, shown in the top right box of Figure 3, captures good uncertainties, particularly on interpolation, but is under-confident for extrapolation.

3.2. Image Classification Datasets

Next, we test the proposed method on state-of-the-art convolutional networks on CIFAR datasets. Similarly to 3.1, we first train an SWA solution, and construct a 5-dimensional random and a 5-dimensional PCA subspaces around it. We also construct a 2-dimensional curve subspace by connecting our SWA solution to another independently trained SWA solution. We visualize the samples from ESS in each of the subspaces in Figure 2. For VI we also visualize the 3σ-region of the closed-form approximate posterior in the random and PCA subspaces. As we can see, the proposed method is able to capture the shape of the posterior in each of the subspaces.

In the PCA subspace we also visualize the SWAG approximate posterior distribution. SWAG overestimates the variance along the first principle component of the SGD trajectory (horizontal axis; see also Figure 5 in (Maddox et al., 2019)). VI is able to fix this issue and provides a better fit for the posterior distribution.

We report the accuracy and negative log-likelihood for each of the subspaces in Table 1. As expected, the performance in the random subspace is the worst, inference in PCA subspace leads to strong performance, and we get the best results in the curve subspace. In the remaining experiments we use the PCA subspace, as it leads to strong performance, and is much cheaper to obtain compared to the curve subspace.

Table 1: Negative log-likelihood and Accuracy for PreResNet-164 for 10-dimensional random, 10-dimensional PCA, and 2-dimensional curve subspaces.

	Random	PCA	Curve
NLL	0.6858 ± 0.0052	0.6652 ± 0.004	0.6464
Accuracy (%)	80.17 ± 0.03	80.54 ± 0.13	81.28

We next apply ESS and simple VI in the PCA subspace on VGG-16, PreResNet-164 and WideResNet28x10 on CIFAR-10 and CIFAR-100. We report the results in Tables 8, 9. Subspace inference is competitive with SWAG and outperforms most of the other baselines, including MC-dropout (Gal and Ghahramani, 2016), temperature scaling (Guo et al., 2017) and KFAC-Laplace Ritter et al. (2018) across the board.

4. Conclusion

We proposed a new approach to Bayesian inference with neural networks: to approximate the posterior distribution over the low-dimensional subspace of the parameter space. The performance of the proposed method crucially depends on the diversity of models within the subspace. In particular, we have demonstrated that simple linear subspaces based on the SGD trajectory contain enough variance for strong Bayesian model averaging, often out-performing full parameter space inference techniques. In future work we plan to design scalable methods for constructing rich subspaces for approximate inference. Being able to perform approximate Bayesian inference in low-dimensional subspaces of DNN parameter space that contain diverse models is a step towards automatic and interpretable Bayesian deep learning.

Acknowledgements

WM, PI, PK and AGW were supported by an Amazon Research Award, Facebook Research, and NSF IIS-1563887. WM was additionally supported by an NSF Graduate Research Fellowship under Grant No. DGE-1650441.

References

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. (2018). Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*.

Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. (2016). Deep gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*, pages 1472–1481.

Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real nvp. In *ICLR*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7587–7597.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, volume abs/1802.10026.

Geyer, C. J. (1991). Markov chain monte carlo maximum likelihood.

Ghashami, M., Liberty, E., Phillips, J. M., and Woodruff, D. P. (2016). Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792.

Guhaniyogi, R. and Dunson, D. B. (2015). Bayesian compressed regression. *Journal of the American Statistical Association*, 110(512):1500–1514.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

Gur-Ari, G., Roberts, D. A., and Dyer, E. (2019). Gradient descent happens in a tiny subspace. *https://openreview.net/forum?id=ByeTHsAqtX*.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

Huggins, J., Campbell, T., and Broderick, T. (2016). Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. In *Uncertainty in Artificial Intelligence*.

Karaletsos, T., Dayan, P., and Ghahramani, Z. (2018). Probabilistic meta-representations of neural networks. arXiv preprint arXiv:1810.00555.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018a). Measuring the intrinsic dimension of objective landscapes. In *ICLR*, page 23.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018b). Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems*. arXiv: 1712.09913.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., and Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. *arXiv preprint arXiv:1902.02476*.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic Gradient Descent as Approximate Bayesian Inference. *JMLR*, 18:1–35.

Minka, T. P. (2001). Automatic choice of dimensionality for pca. In *Advances in neural information processing systems*, pages 598–604.

Murray, I., Prescott Adams, R., and MacKay, D. J. (2010). Elliptical slice sampling. In *Artificial Intelligence and Statistics*.

Neal, R. M. (1996a). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Neal, R. M. (1996b). Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366.

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

Patra, S. and Dunson, D. B. (2018). Constrained bayesian inference through posterior projections. *arXiv preprint arXiv:1812.05741*.

Pradier, M. F., Pan, W., Yao, J., Ghosh, S., and Doshi-Velez, F. (2018). Latent projection bnns: Avoiding weight-space pathologies by learning latent representations of neural network weights. arXiv preprint arXiv:1811.07006.

Ritter, H., Botev, A., and Barber, D. (2018). A scalable laplace approximation for neural networks. In *ICLR*.

Salimbeni, H., Cheng, C.-A., Boots, B., and Deisenroth, M. (2018). Orthogonally decoupled variational gaussian processes. In *Advances in Neural Information Processing Systems*, pages 8725–8734.

Silva, R. and Kalaitzis, A. (2015). Bayesian inference via projections. *Statistics and Computing*, 25(4):739–753.

Vaart, A. W. v. d. (1998). *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge.

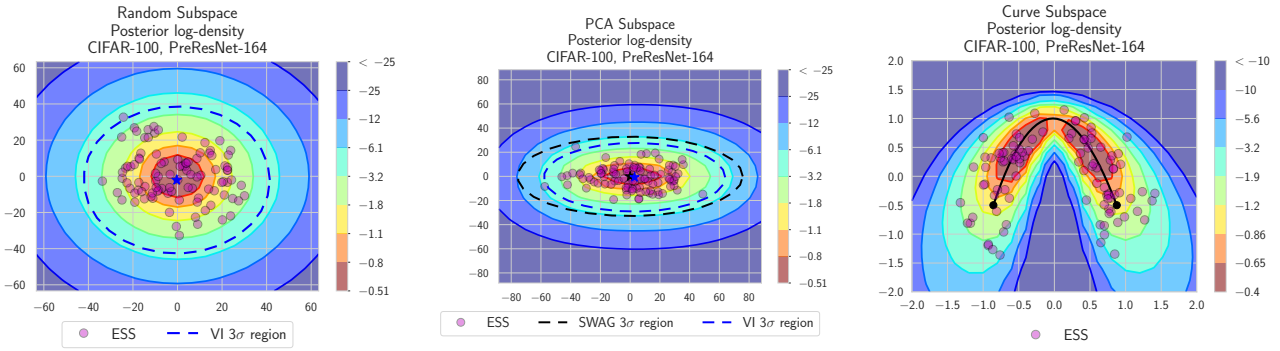
Watanabe, S. (2007). Almost all learning machines are singular. In *2007 IEEE Symposium on Foundations of Computational Intelligence*, pages 383–388. IEEE.

Watanabe, S. (2013). A widely applicable bayesian information criterion. *Journal of Machine Learning Research*, 14(Mar):867–897.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378.

Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernández-Lobato, J. M., and Gaunt, A. L. (2019). Fixing variational bayes: Deterministic variational inference for bayesian neural networks. arXiv preprint arXiv:1810.03958.

Yang, Z., Wilson, A., Smola, A., and Song, L. (2015). A la carte–learning fast kernels. In *Artificial Intelligence and Statistics*, pages 1098–1106.



(a) (b) (c)

Figure 2: Posterior log-density surfaces, ESS samples (shown with magenta circles), and VI approximation posterior distribution (3σ-region shown with blue dashed line) in **(a)** random, **(b)** pca and **(c)** curve subspaces for PreResNet-164 on CIFAR-100. In panel (b) the dashed black line shows the 3σ-region of the SWAG predictive distribution.

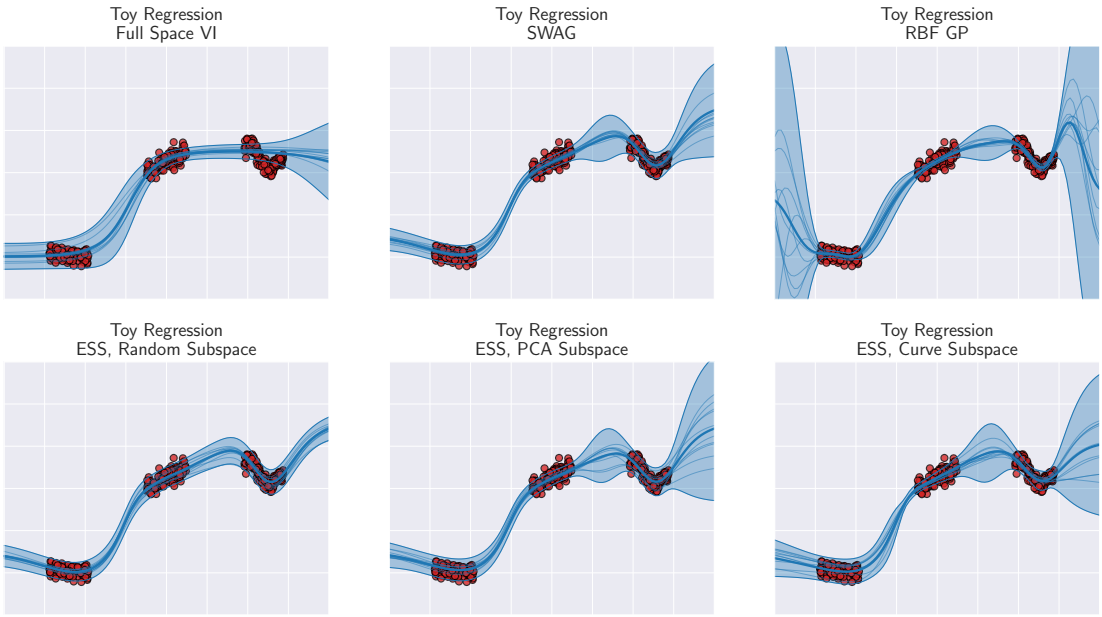


Figure 3: Predictive variances across subspaces for toy regression experiment. Remarkably, elliptical slice sampling with either a PCA or curve subspace captures predictive variances better than full dimensionality variational inference or SWAG. The predictive variances are also in line with a fitted RBF kernel GP for interpolation, but less under-confident for extrapolation.

A. Related Work

Maddox et al. (2019) proposed SWAG, a procedure for approximate Bayesian inference that stores a partial trajectory of the SGD iterates, forming a low-rank plus diagonal covariance matrix for DNN's parameters. They successfully used a Gaussian posterior approximation with this covariance for Bayesian model averaging, producing strong accuracy and uncertainty quantification results on CIFAR and ImageNet datasets. The low-rank part of the SWAG covariance defines a distribution over a low-dimensional subspace spanned by the first principal components of the SGD iterates. We discuss doing inference in this subspace in Section C.2.

Silva and Kalaitzis (2015) consider the related problem of Bayesian inference using projected methods for constrained latent variable models, with applications to probabilistic PCA.

Pradier et al. (2018) proposes to perform variational inference (VI) in a subspace formed by an auto-encoder trained on a set of weights of independently trained models; however, the method requires many models to train the auto-encoder, limiting scalability. Similarly, Karaletsos et al. (2018) propose to use a meta-prior in low dimensional space to perform variational inference for BNNs.

Patra and Dunson (2018) give several theoretical guarantees for Bayesian inference with posterior projections. For instance, they show that if a constrained posterior density can be induced with a minimal distance mapping from an unconstrained posterior density, then there must also exist a prior density on the constrained parameter space that produces the constrained posterior density. As a result, the proposed method is to approximately sample from the unconstrained posterior before using a minimal distance mapping into the constrained parameter space. In many respects, this provides theoretical motivation to our method if we view SGD as an approximate sampling algorithm, i.e. Mandt et al. (2017); the parameters are then constrained to lie in a subspace, i.e. the first 10 components of the SGD trajectory.

Bayesian coresets (Huggins et al., 2016) and Bayesian compressed regression (Guhaniyogi and Dunson, 2015) can be viewed as subspace inference techniques, but in data space rather than in parameter space. That is, these methods attempt to find a projection of the data into lower dimensions before performing inference, whereas the proposed method uses the full data and attempts to project the model into a lower dimensional space.

B. Computational Benefits

For a thought experiment on the potential benefits from performing black-box MCMC (i.e. no structure exploitation), we know (see Section 4.4 of Neal et al. (2011)) that the average time (number of likelihood and gradient evaluations) for an independent sample to be drawn using Hamiltonian Monte Carlo (HMC) grows as $p^{5/4}$. Similarly, the time for an independent sample Metropolis-Hastings grows as p^2.

Reducing the dimensionality of the problem should enable much faster mixing of chains in the MCMC setting. Even if the dimensionality of the subspace grew as $K = \log p$, then the time per independent sample using HMC would be $(\log p)^{5/4}$, rather than $p^{5/4}$, a significant speed-up. Of course, this thought experiment ignores the time necessary for likelihood computations and possible issues with the behavior of the projected likelihood.

C. Subspace Construction

In the last section we showed how to perform inference in a given subspace $\mathcal{S}$. We now discuss various ways to construct $\mathcal{S}$. For each of the subspaces, we also discuss the choice of prior distribution within the subspace.

C.1. Random Subspaces

The simplest approach to construct a subspace, $\mathcal{S}$, is to choose it randomly. To do so we draw K random $v_1, \ldots, v_K \sim \mathcal{N}(0, I_p)$ in the weight space. We then rescale each of the vectors to have norm 1. Random subspaces are quick to generate and form, but contain no information about the model. Note that this is exactly the approach Li et al. (2018a) used to attempt to train networks from scratch, requiring much higher dimensions than are considered in this paper. To include some network information, we use the SWA (Izmailov et al., 2018) solution as the shift vector $\hat{w}$ of the subspace definition in Eq. (1).

For the prior within a random subspace we use $\mathcal{N}(0, \sigma^2 \cdot I_K)$, which induces a linear combination of chi-square random variables as a prior in weight space.

C.2. PCA of the SGD trajectory

Intuitively we want the subspace $\mathcal{S}$ that we do inference in to (1) contain a diverse (producing meaningfully different predictions on test data) set of models and (2) be cheap to construct. (Garipov et al., 2018; Izmailov et al., 2018) argue that the subspace spanned by the SGD trajectory satisfies both (1) and (2). They run SGD starting from a pre-trained solution with a

high constant learning rate and then ensemble predictions or average the weights of the iterates. Further, (Maddox et al., 2019) showed that fitting the SGD iterates with a Gaussian distribution with a low-rank plus diagonal covariance and then using this Gaussian for Bayesian model averaging it is possible to obtain well-calibrated uncertainty estimates in computer vision problems. These observations motivate inference in the subspace spanned by the SGD trajectory.

Similarly to Li et al. (2018b); Maddox et al. (2019) we propose using the first few PCA components of the SGD trajectory as basis directions v_i for the subspace. Following (Garipov et al., 2018; Izmailov et al., 2018; Maddox et al., 2019) we run SGD with a high constant learning rate from a pre-trained solution and capture snapshots w_i of weights at the end of each of T epochs. We use the SWA solution (the mean of the iterates $w_{\text{SWA}} = \frac{1}{T}\sum_i w_i$) as the shift vector $\hat{w}$ in (1). We also store the deviations $v_i = w_{\text{SWA}} - w_i$ for the last M epochs. The number M here is determined by the amount of memory we can use, and is set to $M = 20$ in the experiments. We then run PCA based on randomized SVD (Halko et al., 2011)[3] on the matrix $\hat{D}$ comprised of vectors $w_1, \ldots, w_M$ and use the first K principal components $v_1, \ldots, v_K$ to define the subspace (1). We assume here that the vectors v_i are have norms proportional to the singular values of the matrix $\hat{D}$.

This procedure can be summarized in Algorithm 2.

For the prior in the subspace we use $\mathcal{N}(0, I)$. This way the prior distribution attempts to match the distribution of the SGD iterates, inducing a low rank version of the SWAG approximation (Maddox et al., 2019). Note however, that the SWAG distribution is defined over the full parameter space and uses a low-rank plus diagonal covariance, while our prior is only defined over a subspace, and only includes the low-rank covariance term from SWAG. Further, SWAG uses a slightly different procedure for computing the deviation vectors v_i, see Maddox et al. (2019) for details.

As generally the amount of weight snapshots we can store in memory is limited, in the PCA subspace we only use the last M epochs of SGD trajectory. To remedy this issue and keep the size of the subspace small, we could use any online PCA technique instead: one promising approach is frequent directions (Ghashami et al., 2016).

[3]Implemented in sklearn.decomposition.TruncatedSVD.

Algorithm 2 Subspace Construction with PCA of SGD trajectory

w_0: pretrained weights; η: learning rate; T: number of steps; c: moment update frequency; M: maximum number of columns in deviation matrix; K: rank of PCA approximation

Train SWAG

- $\overline{w} \leftarrow w_0$ {Initialize mean}
- **for** $i \leftarrow 1, 2, ..., T$ **do**
  - $w_i \leftarrow w_{i-1} - \eta \nabla_w \mathcal{L}(w_{i-1})$ {SGD update}
  - **if** $\texttt{MOD}(i, c) = 0$ **then**
    - $n \leftarrow i/c$ {Number of models}
    - $\overline{w} \leftarrow \frac{n\overline{w} + w_i}{n+1}$ {Update mean}
    - **if** $\texttt{NUM\_COLS}(\bar{D}) = M$ **then**
      - $\texttt{REMOVE\_COL}(\bar{D}[:, 1])$
    - $\texttt{APPEND\_COL}(\bar{D}, w_i - \overline{w})$ {Store deviation}
  - $U, S, V^T \leftarrow SVD(\bar{D})$ {Truncated SVD}
- **return** $w_{\text{SWA}} = \overline{w}, \quad D = SV^T$

C.3. Curve Subspaces

Garipov et al. (2018) proposed a method to find paths of near-constant low loss (and consequently high posterior density) in the weight space between given independently trained networks. These curves lie in 2-dimensional subspaces of the weight space. We visualize the loss surface in such space on a synthetic regression problem in Figure 1 (d). We explore doing inference in the 2-dimensional subspace containing such a curve, which we refer to as *curve subspace*. While using the curve subspace is of limited practicality (constructing the space requires training two independent networks and then fitting the curve), it serves as an example of a rich subspace containing a large number of diverse solutions. It also serves as a gauge of the ability of sampling procedures in the subspace to find regions of high posterior density when the posterior is very non-Gaussian.

We use the point $\mu = (w_0 + w_{1/2} + w_1)/3$ as the mean of the normal prior distribution, where w_0 and w_1 are the endpoints, and $w_{1/2}$ is the midpoint of the curve. We use a covariance $\sigma^2 I$, where $\sigma = \|\mu - w_0\|$.

D. Automatic Dimensionality Choice

A critical hyper-parameter in the choice of subspace is the selection of the subspace size, . In this section, we describe how to automatically set this dimensionality when using Frequent Directions for online low rank covariance approximation. Minka (2001) utilized a Laplace approximation to the model evidence of a probabilistic model for PCA. This probabilistic model

can be written as

$$p(x|H, v) \sim N(0, HH^\top + vI),$$

where

$$p(e) \sim N(0, vI_d) \quad \text{and} \quad p(w) \sim N(0, I).$$

Typically, the maximum likelihood estimate for $\hat{H}$ can be written as

$$\hat{H} = U(\Lambda - \hat{v}I_k)^{1/2}R,$$

where U is orthogonal containing the top eigenvectors of the covariance and $\hat{v}$ is the maximum likelihood estimate of the variance, given by:

$$\hat{v} = \frac{\sum_{i=k+1}^{p} \lambda_i}{p-k} = \frac{tr(\Sigma - \Sigma_k)}{p-k} \tag{4}$$

When the data covariance can be stored in memory, $\hat{H}$ is just the rank k truncated SVD subtracted by $\hat{v}I$ and $\hat{v}$ is just a scaled version of the remaining eigenvalues.

After much linear algebra, a Laplace approximation to the model evidence for PCA is given by:

$$p(S|k) \propto p(U) \left[\prod_{j=1}^{k} \lambda_j\right] \hat{v}^{-N(d-k)/2} \tag{5}$$

$$(2\pi)^{(m+k)/2} |A_z|^{-1/2} N^{-k/2}, \tag{6}$$

where

$$p(U) \propto 2^{-k} \prod_{i=1}^{k} \Gamma((d-i+1)/2)\pi^{-(d-i+1)/2},$$

$$|A_z| = \prod_{i=1}^{k} \prod_{j=i+1}^{d} (\hat{\lambda}_j^{-1} - \hat{\lambda}_i^{-1})(\lambda_i - \lambda_j)N,$$

and $m = dk - k(k+1)/2$, the degrees of freedom of U.

Our observations on $p(U)$ found that it grows exponentially with k, the number of dimensions, which is potentially problematic when $k << d$. As a result, we place a stronger regularization term on the model evidence, equivalent to placing a prior on the dimensionality, k, explicitly: adding a term of $N^{-k-3/2m}$ into Eq 6.[4] This yields a marginal likelihood that becomes:

$$p(S|k) \propto p(U) \left[\prod_{j=1}^{k} \lambda_j\right] \hat{v}^{-N(d-k)/2} \tag{7}$$

$$(2\pi)^{(m+k)/2} |A_z|^{-1/2} N^{-\mathbf{1/2(k+m)}}. \tag{8}$$

[4]We developed this term by noticing the dependence on m in Minka's BIC approximation for PCA (Minka, 2001).

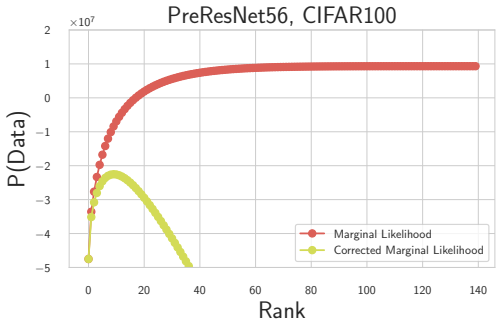


Figure 4: Log marginal likelihood as a function of dimensionality for PreResNet56 on CIFAR100 using both versions of the Laplace approximation. The comparison is similar to Occam's razor versus Occam's hill as described in MacKay (2003).

As this is is a much stronger regularization term, we need to test its efficacy on simulated and test data. To do so, we compared to the *sklearn.decomposition.PCA*[5] implementation of Minka's approximate marginal likelihood for PCA repeating the same trials as in Minka (2001).

E. Eigen-Gaps of the Hessian and Fisher matrices

We can see similar behavior within the eigenvalues of both the Hessian and the empirical Fisher information matrix, at the end of training in Figure 6. To compute these eigenvalues, we used a GPU-enabled Lanczos method in GPyTorch (Gardner et al., 2018) on a pre-trained PreResNet164. We ran Lanczos for 100 steps, estimating 100 eigenvalues, before shifting by the maximum eigenvalue, and running for 200 steps, estimating 200 eigenvalues. Lanczos tends to converge from the "outside in" so to speak, see Chaper 7 of (Demmel, 1997) for theoretical guarantees, so that it ought to be possible to pick up eigen-gaps. As such, we would expect the training dynamics of SGD to primarily use these much larger eigenvalues, a finding shown empirically by Li et al. (2018b); Gur-Ari et al. (2019).

F. Approximate Inference Methods

For inference in the subspace we can use a wide range of approximate Bayesian inference techniques. We consider the following methods.

[5]`https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

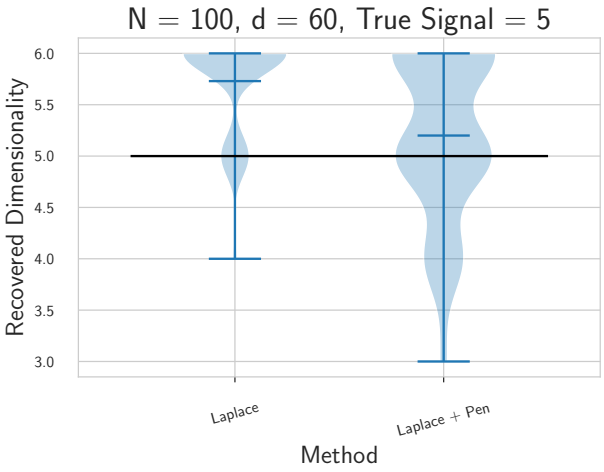


Figure 5: Dimensionality recovery over 100 trials for PCA using estimates of Eq. 8 following experimental setup of Minka (2001). The dimensionality recovered with the secondary penalty performs at least as well as the standard marginal likelihood.

Elliptical Slice Sampling As the dimensionality of the subspace K is not too high, gradient-free methods such as elliptical slice sampling (ESS) (Murray et al., 2010) can be used to sample from the projected posterior distribution. Elliptical slice sampling is designed to have no tuning parameters, and only requires a Gaussian prior in the subspace.[6]

For networks that cannot evaluate all of the training data in memory at a single time, it is easily possible to sum the loss over mini-batches computing a full log probability, without storing gradients.

[6]We use the Python implementation at `https://github.com/jobovy/bovy_mcmc/blob/master/bovy_mcmc/elliptical_slice.py`.

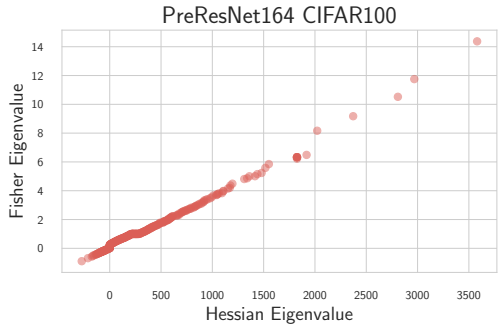


Figure 6: Plot of 300 eigenvalues of the Fisher and Hessian matrices for a PreResNet164 on CIFAR100. A clear separation exists between the top 20 or so eigenvalues and the rest, which are crowded together.

NUTS We also test the No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014), an HMC method (Neal et al., 2011), that dynamically tunes the hyperparameters (step-size and leapfrog steps) of HMC.[7] NUTS has the advantage of being nearly black-box: only a joint likelihood and its gradients need to be defined. However, full gradient calls are required, which can be tricky to cache and a constant factor slower than a full likelihood calculation.

Simple Variational Inference We experiment with variational inference in the subspace using the fully-factoried Gaussian posterior approximation family. Fully-factorized Gaussians are among the simplest and the most common variational families. Unlike ESS or NUTS, VI can be trained with mini-batches, but it is limited in the distributions it can represent.

RealNVP The simple fully-factorized Gaussian family of distributions can be too constrained, so we also experiment with normalizing flows, which parametrize the variational distribution family with invertible neural networks. We adapt the RealNVP method of (Dinh et al., 2017), due to its flexibility and success on image reconstruction and generation tasks.

G. Experiments

G.1. UCI Regression

We next compare subspace inference methods on regression tasks to a variety of methods for approximate Bayesian inference in BNNs. We use the preprocessing and standardization in `https://github.com/hughsalimbeni/bayesian_benchmarks`.

To ensure fair comparison to other methods, we do not assess the Bayesian model averaged estimate of the probability, which would be $\frac{1}{N}\sum_{i=1}^{N} p(y|x, \theta_i) = p(y_i|x_i, \mu(x_i;\theta_i), \sigma^2(x_i;\theta_i))$, where $\theta_i \sim q$. Instead, we compute a sample statistics based estimator and moment match a fitted Gaussian,[8] computing $p(y|\hat{\mu}, \hat{\sigma^2})$, where $\hat{\mu}(x_i) = \frac{1}{N}\sum_{i=1}^{N} \mu(x_i;\theta_i)$ and $\hat{\sigma}^2(x_i) = \frac{1}{N}\sum_{i=1}^{N}(\sigma^2(x_i;\theta) + \mu(x_i;\theta)^2) - \hat{\mu}(x_i)^2$.

In all experiments, we replicated over 10 trials reserving 90% of the data for training and the other 10% for testing, following the set-up of Bui et al. (2016) and Wilson et al. (2016).

[7]Implemented in Pyro (Bingham et al., 2018).

[8]This is the same estimator used in Wu et al. (2019), and is closer to how a BNN would be used in practice.

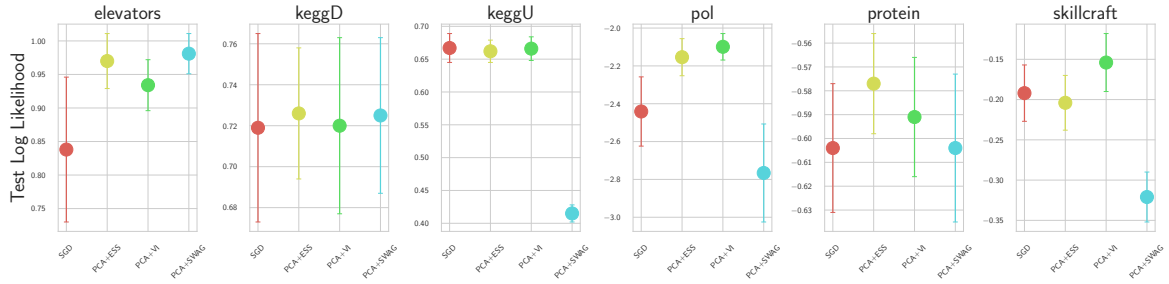


Figure 7: Test log-likelihoods for proposed methods on six UCI regression datasets. Inference with a subspace, particularly using PCA and inferring the posterior with elliptical slice sampling is associated with higher test log-likelihoods, outperforming SGD and SWAG.

G.1.1. Large Scale UCI Regression Datasets

Here, on all models except skillcraft [9], like in Wilson et al. (2016), we use a feed forward network with five hidden layers: [1000, 1000, 500, 50, 2, 2], ReLU activations and heteroscedastic noise variance. Noting that Gaussian processes typically include a noise variance term into the kernel matrix, we additionally learn a global noise variance, so that the variance of a given point is given by: $\sigma^2(x) = s^2 + \sigma_\theta^2(x)$, where $\sigma_\theta^2(x)$ is the variance output from the final layer of the network. We use softplus parameterizations to ensure positiveness of the variance, initializing the global variance at $s^2 = 1$ (the total variance in the dataset).

On large-scale UCI regression datasets, we compare to two types of approximate Gaussian processes: orthogonally decoupled variational Gaussian Processes (OrthVGP, Salimbeni et al. (2018)) and Fastfood approximate kernels (FF, Yang et al. (2015), results from Wilson et al. (2016))as well as deep kernel learning with a spectral mixture kernel (DKL, Wilson et al. (2016)).

For SGD trained networks and subspace models, we additionally report the predictive calibration of the methods. The test log-likelihoods and RMSEs are shown in Table 6 and 5, while the coverage of the 95% predictive intervals are shown in Table 7. We summarize the test log-likelihoods shown in Figure 7, where it is possible to see that even when predicting with statistics, the test log-likelihood typically decreases in comparison to the SGD model. Finally, we plot the coverage of the 95% predictive intervals in Figure 8.

For the large-scale UCI regression tasks, we manually tuned hyper-parameters (batch size, learning rate, and epochs) to match the SGD DNN results in Table 1 of Wilson et al. (2016). Here, there is one significant difference which is that our networks use heteroscedastic uncertainty, while those networks use homoscedastic uncertainty (a fixed variance). However, we found that the results were similar in terms of RMSE, but fitting networks with heteroscedastic uncertainty allows for a principled comparison of test log-likelihood and calibration.

We additionally tried fitting models without a global variance parameter, but found that they were typically more over-confident than models with a global variance parameter.

Following Wilson et al. (2016), for the UCI regression tasks with more than 6,000 data points, we used networks with the following structure: [1000, 1000, 500, 50, 2], while for skillcraft, we used a network with: [1000, 500, 50, 2]. We used a learning rate of $1e-3$, doubling the learning rate of bias parameters, a batch size of 400, momentum of 0.9, and weight decay of $4e-3$, training for 200 epochs. For skillcraft and pol, we only trained for 100 epochs, while for skillcraft we used a learning rate of $5e-4$ and for keggD, we used a learning rate of $1e-4$. We additionally used a subspace prior of 1.0.

In Table 5, we report RMSE results compared to two types of approximate Gaussian processes (Salimbeni et al., 2018; Yang et al., 2015); note that the results for OrthVGP are reproduced from Appendix Table F of Salimbeni et al. (2018) but scaled by the standard deviation of the respective dataset.

We repeated each model over 10 random train/test splits; each test set consisted of 10% of the full dataset. All data was pre-processed to have mean zero and variance one.

G.1.2. Small UCI Regression Datasets

In this set of experiments, we compare to the state-of-the-art approximate BNN inference methods including deterministic variational inference (DVI) (Wu et al.,

[9] For skillcraft, we use a smaller architecture [1000, 500, 50, 2, 2].

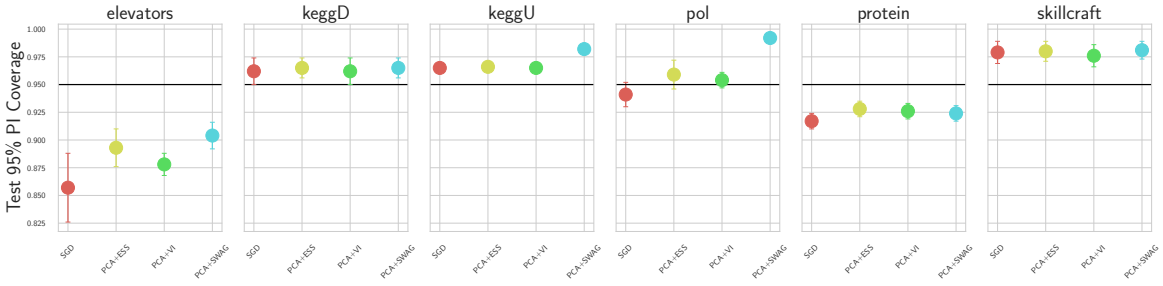


Figure 8: Coverage of 95% prediction interval for models trained on six UCI datasets. In nearly all cases, using a subspace inference method is associated with closer to 95% coverage than for models trained using only SGD.

Table 2: Unnormalized test log-likelihoods on small UCI datasets for proposed methods, as well as direct comparisons to the numbers reported in deterministic variational inference (DVI, Wu et al. (2019)) and Deep Gaussian Processes with expectation propagation (DGP1-50, Bui et al. (2016)), and variational inference (VI) with the re-parameterization trick (Kingma et al., 2015).

dataset	N	D	SGD	PCA+ESS	PCA+VI	PCA+SWAG	DVI	DGP1-50	VI
boston	506	13	-2.712 ± 0.258	-2.742 ± 0.275	-3.083 ± 0.460	-2.755 ± 0.285	**-2.41** ± 0.02	-2.33 ± 0.06	-2.43 ±0.03
concrete	1030	8	-3.207 ± 0.183	-3.162 ± 0.165	-3.241 ± 0.203	-3.186 ± 0.176	**-3.06** ± 0.01	-3.13 ± 0.03	-3.04 ±0.02
energy	768	8	-1.736 ± 1.613	-1.563 ± 1.243	-1.715 ± 1.588	-1.679 ± 1.488	**-1.01** ± 0.06	-1.32 ± 0.03	-2.38 ±0.02
naval	11934	16	6.567 ± 0.185	6.541 ± 0.095	**6.708** ± 0.105	6.708 ± 0.105	6.29 ± 0.04	3.60 ± 0.33	5.87 ±0.29
yacht	308	6	-0.418 ± 0.426	**-0.225** ± 0.400	-0.396 ± 0.419	-0.404 ± 0.418	-0.47 ± 0.03	-1.39 ± 0.14	-1.68 ±0.04

Table 3: RMSE on small UCI datasets. Subspace inference typically performs comparably to SGD and SWAG.

	SGD	PCA+ESS	PCA+VI	SWAG
boston	3.856 ± 1.027	3.779 ± 1.034	**3.665** ± 1.028	3.742 ± 1.036
concrete	5.279 ± 0.400	5.263 ± 0.396	**5.232** ± 0.387	5.250 ± 0.389
energy	1.602 ± 0.275	1.598 ± 0.274	**1.587** ± 0.272	1.594 ± 0.273
naval	**0.001** ± 0.000	**0.001** ± 0.000	**0.001** ± 0.000	**0.001** ± 0.000
yacht	0.973 ± 0.374	**0.972** ± 0.375	0.973 ± 0.375	0.973 ± 0.375

2019), single-layer deep GPs (DGP) with expectation propagation (Bui et al., 2016), and re-parameterization VI (Kingma and Welling, 2013). In these experiments, we follow the set-up of (Wu et al., 2019) and use a single hidden layer with 50 units and also model the variance with heteroscedasticity. The test log-likelihoods, RMSEs and test calibration results are presented in Tables 2, 3 and 4.

In Tables 2 and 3, we can see that the subspace methods outperform SGD by a small margin on both RMSE and test log likelihood and are very competitive with DVI and deep GPs. In Table 4, we see that the subspace methods are typically much better calibrated than their SGD and SWAG counterparts.

For the small UCI regression datasets, we use the architecture from Wu et al. (2019) with one hidden layer with 50 units. We use Adam optimizer, manually tune learning rate and weight decay, and use batch size of $N/10$ where N is the dataset size. All models predict heteroscedastic uncertainty (i.e. output a variance). In Table 2, we compare subspace inference methods to deterministic VI (DVI, Wu et al. (2019)) and deep Gaussian processes with expectation propagation (DGP1-50 Bui et al. (2016)). ESS and VI in the PCA subspace outperform DVI on two out of five datasets.

G.2. Image Classification Results

For the experiments on CIFAR datasets we are following the framework of (Maddox et al., 2019). We report the negative log-likelihood and accuracy for our method and baselines in Tables 8 and 9.

G.3. Automatic Choice of Dimensionality

In Figure 9, we show the eigenvalues of the trajectory computed using randomized SVD as in Section C.2, where it is possible to see that the eigenvalues of the trajectory decay extremely quickly (from an initial eigenvalue above 100). Heuristically, this sug-

Table 4: Calibration on small-scale UCI datasets. Bolded numbers are those closest to 95% predicted coverage).

	N	D	SGD	PCA+ESS	PCA+VI	SWAG
boston	506	13	**0.882** ± 0.042	0.863 ± 0.049	0.802 ± 0.042	0.860 ± 0.045
concrete	1030	8	0.858 ± 0.043	**0.859** ± 0.039	0.849 ± 0.042	0.857 ± 0.040
energy	768	8	0.947 ± 0.026	0.953 ± 0.027	**0.949** ± 0.027	**0.951** ± 0.027
naval	11934	16	**0.948** ± 0.051	0.978 ± 0.006	0.967 ± 0.008	0.967 ± 0.008
yacht	308	6	0.895 ± 0.069	**0.948** ± 0.040	0.898 ± 0.067	0.898 ± 0.067

Table 5: RMSE comparison amongst methods on larger UCI regression tasks, as well as direct comparisons to the numbers reported in deep kernel learning with a spectral mixture kernel (DKL, (Wilson et al., 2016)), orthogonally decoupled variational GPs (OrthVGP, Salimbeni et al. (2018)), and FastFood kernel GPs (FF, Yang et al. (2015) from Wilson et al. (2016)). Subspace based inference typically outperforms SGD and approximate GPs and is competitive with DKL.

					w				
dataset	N	D	SGD	PCA+ESS	PCA+VI	SWAG	DKL	OrthVGP	FF
elevators	16599	18	0.092 ± 0.003	0.090 ± 0.002	0.090 ± 0.002	0.090 ± 0.002	$\mathbf{0.084} \pm 0.02$	0.0952	0.089 ± 0.002
keggD	48827	20	0.121 ± 0.003	0.122 ± 0.003	0.123 ± 0.003	0.134 ± 0.005	$\mathbf{0.10} \pm 0.01$	0.1198	0.12 ± 0.00
keggU	63608	27	0.125 ± 0.024	0.125 ± 0.023	0.125 ± 0.023	0.125 ± 0.023	$\mathbf{0.11} \pm 0.00$	0.1172	0.12 ± 0.00
protein	45730	9	0.443 ± 0.009	$\mathbf{0.440} \pm 0.007$	0.444 ± 0.009	0.447 ± 0.011	0.46 ± 0.01	0.46071	0.47 ± 0.01
skillcraft	3338	19	0.284 ± 0.015	0.286 ± 0.016	0.276 ± 0.015	0.322 ± 0.015	$\mathbf{0.25} \pm 0.00$		$\mathbf{0.25} \pm 0.02$
pol	15000	26	3.018 ± 0.310	2.816 ± 0.196	$\mathbf{2.705} \pm 0.181$	3.289 ± 0.408	3.11 ± 0.07	6.61749	4.30 ± 0.2

Table 6: Normalized test log-likelihoods on larger UCI datasets. Subspace methods outperform an approximate GP approach (OrthVGP) and SGD, typically often out-performing SWAG.

dataset	N	D	SGD	PCA+ESS	PCA+VI	SWAG	OrthVGP
elevators	16599	18	-0.538 ± 0.108	-0.406 ± 0.041	-0.442 ± 0.038	**-0.395** ± 0.030	-0.4479
keggD	48827	20	0.985 ± 0.022	0.981 ± 0.017	0.984 ± 0.018	0.734 ± 0.013	**1.0224**
keggU	63608	27	0.700 ± 0.046	$\mathbf{0.707} \pm 0.032$	0.702 ± 0.043	$\mathbf{0.707} \pm 0.038$	0.7007
protein	45730	9	-0.861 ± 0.027	**-0.834** ± 0.021	-0.849 ± 0.025	-0.861 ± 0.031	-0.9138
skillcraft	3338	19	-1.147 ± 0.035	-1.159 ± 0.034	**-1.109** ± 0.036	-1.276 ± 0.031	
pol	15000	26	1.290 ± 0.1834	1.577 ± 0.098	$\mathbf{1.633} \pm 0.070$	0.965 ± 0.259	0.1586

Table 7: Calibration on large-scale UCI datasets. Bolded numbers are those closest to 95% predicted coverage).

dataset	N	D	SGD	PCA+ESS	PCA+VI	SWAG
elevators	16599	18	0.857 ± 0.031	0.893 ± 0.017	0.878 ± 0.010	$\mathbf{0.904} \pm 0.012$
keggD	48827	20	$\mathbf{0.965} \pm 0.002$	0.966 ± 0.003	$\mathbf{0.965} \pm 0.003$	0.982 ± 0.003
keggU	63608	27	$\mathbf{0.962} \pm 0.012$	0.965 ± 0.009	$\mathbf{0.962} \pm 0.012$	0.965 ± 0.009
protein	45730	9	0.917 ± 0.007	$\mathbf{0.928} \pm 0.007$	0.926 ± 0.007	0.924 ± 0.007
skillcraft	3338	19	0.979 ± 0.010	0.980 ± 0.009	$\mathbf{0.976} \pm 0.010$	0.981 ± 0.008
pol	15000	26	$0.941 \pm 0.011)$	0.959 ± 0.013	$\mathbf{0.954} \pm 0.007$	0.992 ± 0.004

Table 8: NLL for various versions of subspace inference, SWAG, temperature scaling, and dropout.

Dataset	Model	PCA + VI	PCA + ESS	SWA	SWAG	KFAC-Laplace	SWA-Dropout	SWA-Temp
CIFAR-10	VGG-16	0.2052 ± 0.0029	0.2068 ± 0.0029	0.2621 ± 0.0104	$\mathbf{0.2016} \pm 0.0031$	0.2252 ± 0.0032	0.2328 ± 0.0049	0.2481 ± 0.0245
CIFAR-10	PreResNet-164	0.1247 ± 0.0025	0.1252 ± 0.0018	0.1450 ± 0.0042	$\mathbf{0.1232} \pm 0.0022$	0.1471 ± 0.0012	0.1270 ± 0.0000	0.1347 ± 0.0038
CIFAR-10	WideResNet28x10	0.1081 ± 0.0003	0.1090 ± 0.0038	0.1075 ± 0.0004	0.1122 ± 0.0009	0.1210 ± 0.0020	0.1094 ± 0.0021	$\mathbf{0.1064} \pm 0.0004$
CIFAR-100	VGG-16	0.9904 ± 0.0218	1.015 ± 0.0259	1.2780 ± 0.0051	$\mathbf{0.9480} \pm 0.0038$	1.1915 ± 0.0199	1.1872 ± 0.0524	1.0386 ± 0.0126
CIFAR-100	PreResNet-164	$\mathbf{0.6640} \pm 0.0025$	0.6858 ± 0.0052	0.7370 ± 0.0265	0.7081 ± 0.0162	0.7881 ± 0.0025		0.6770 ± 0.0191
CIFAR-100	WideResNet28x10	$\mathbf{0.6052} \pm 0.0090$	0.6096 ± 0.0072	0.6684 ± 0.0034	0.6078 ± 0.0006	0.7692 ± 0.0092	0.6500 ± 0.0049	0.6134 ± 0.0023

gests that the interesting directions in the trajectory are well-described in only a very few dimensions. To automatically choose the dimensionality, we attempted to apply the Laplace approximation to the probabilistic view of SVD as in Minka (2001) - see Appendix 6 for the equation; however, this failed to reduce the dimensionality. This failure is explained the marginal likelihood was estimated using biased estimates of the trajectory eigenvalues[10] and the exponential growth of the prior on the eigenvector matrix with respect to dimensionality.

However, a simple modification to impose a prior on the rank, K, gave maximized likelihoods in 10 dimensions on a PreResNet 56 on CIFAR100; the likelihood as a function of rank is shown in Figure 4. Replicating the prior experiments, we trained for 325 epochs but stored 140 copies of the model in the covariance matrix. This modification is given in Appendix Eq. 8, while its reasonableness on a toy problem is given in Figure 5.

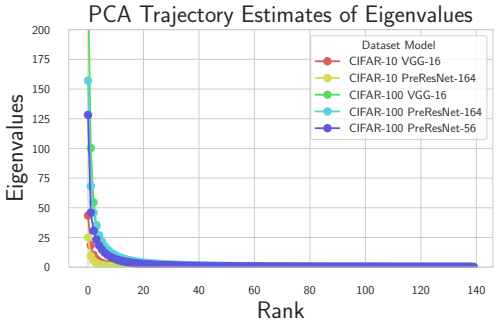


Figure 9: Eigenvalues of trajectory covariance estimated from randomized SVD across three architectures on CIFAR-10 and CIFAR-100. The trajectory decays extremely quickly, decaying towards 0 around 10-20 setps.

H. Additional Toy Regression Plots

In Figure 10 we present the predictive disribution plots for all the inference methods and subspaces. We additionally visualize the samples over poterior density surfaces for each of the methods in Figure 11.

[10]The bounds on reconstruction error for randomized rank-k SVD are in terms of the $k+1$-th singular value (see Theorem 1.1 of Halko et al. (2011)), suggesting that the k singular values estimated might be biased, and both the variance estimate and resulting eigenvalues will be biased.

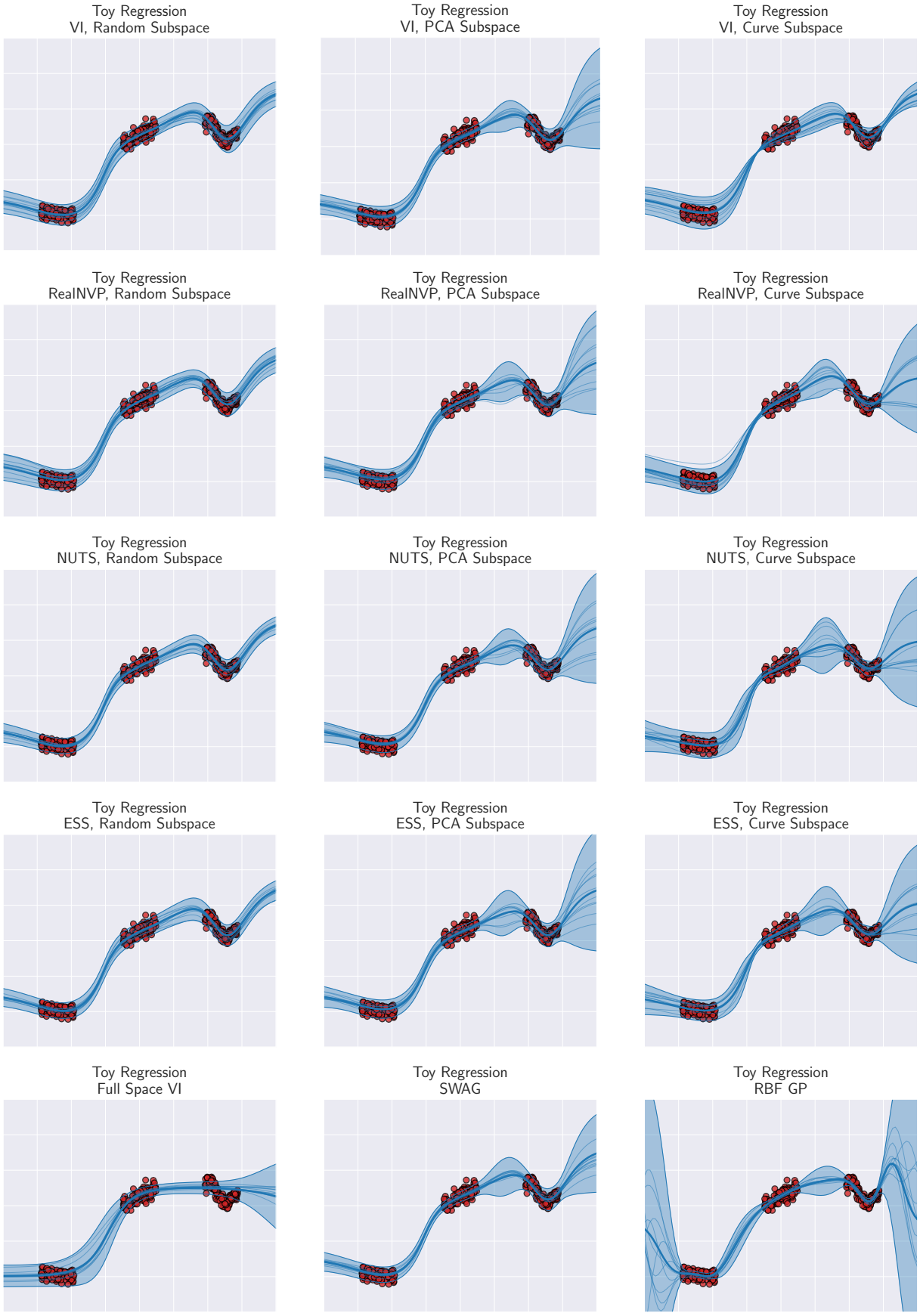


Figure 10: Toy regression predictive distributions across inference methods and subspaces.

Table 9: Accuracy for various versions of subspace inference, SWAG, temperature scaling, and dropout.

Dataset	Model	PCA + VI	PCA + ESS	SWA	SWAG	KFAC-Laplace	SWA-Dropout	SWA-Temp
CIFAR-10	VGG-16	93.61 ± 0.02	$\mathbf{93.66} \pm 0.08$	93.61 ± 0.11	93.60 ± 0.10	92.65 ± 0.20	93.23 ± 0.36	93.61 ± 0.11
CIFAR-10	PreResNet-164	95.96 ± 0.13	95.98 ± 0.09	96.09 ± 0.08	96.03 ± 0.02	95.49 ± 0.06	$\mathbf{96.18} \pm 0.00$	96.09 ± 0.08
CIFAR-10	WideResNet28x10	96.32 ± 0.03	96.38 ± 0.05	$\mathbf{96.46} \pm 0.04$	96.32 ± 0.08	96.17 ± 0.00	96.39 ± 0.09	96.46 ± 0.04
CIFAR-100	VGG-16	$\mathbf{74.83} \pm 0.08$	74.62 ± 0.37	74.30 ± 0.22	74.77 ± 0.09	72.38 ± 0.23	72.50 ± 0.54	74.30 ± 0.22
CIFAR-100	PreResNet-164	80.52 ± 0.18	$\mathbf{80.54} \pm 0.13$	80.19 ± 0.52	79.90 ± 0.50	78.51 ± 0.05		80.19 ± 0.52
CIFAR-100	WideResNet28x10	$\mathbf{82.63} \pm 0.26$	82.49 ± 0.23	82.40 ± 0.16	82.23 ± 0.19	80.94 ± 0.41	82.30 ± 0.19	82.40 ± 0.16

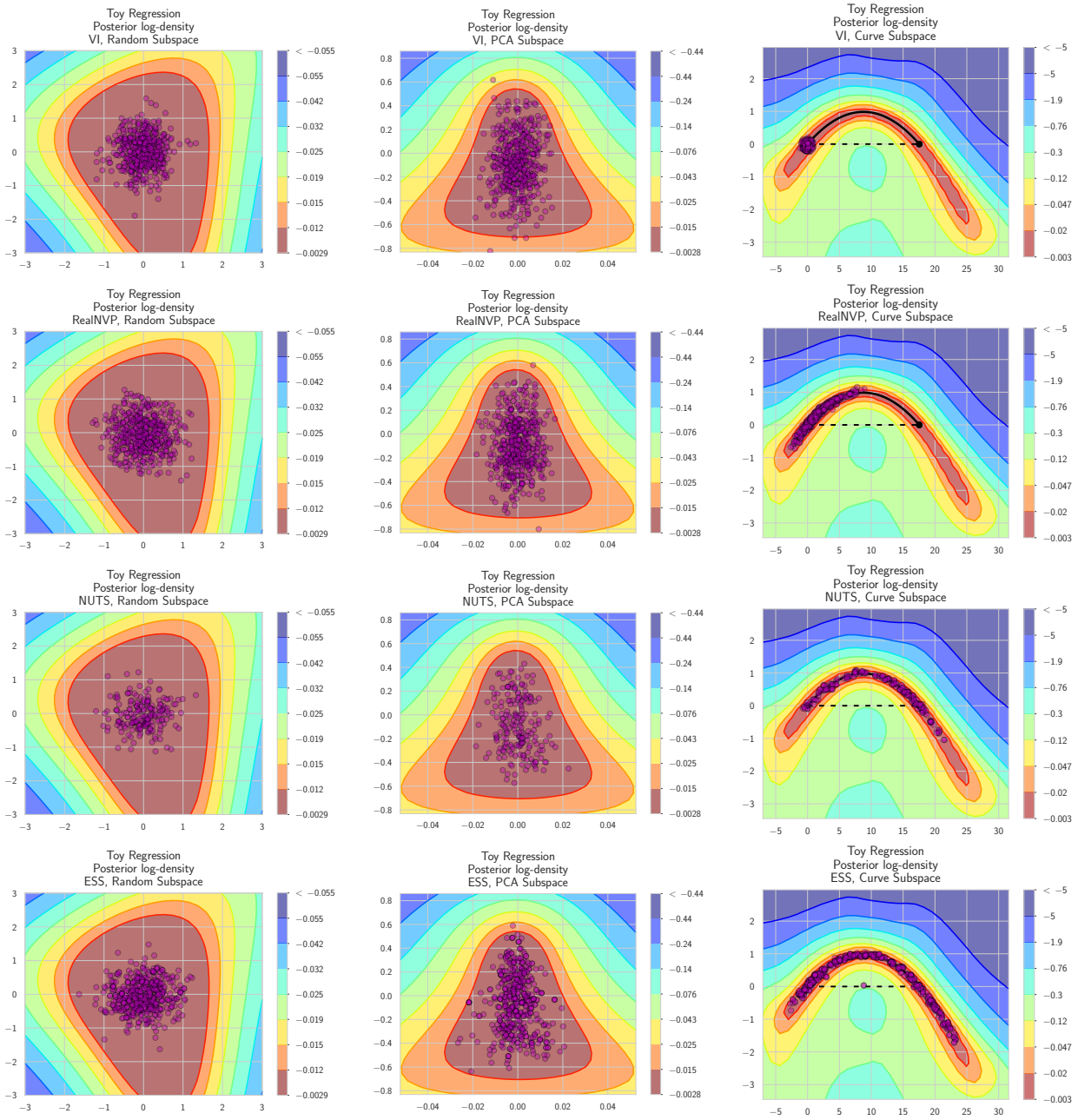


Figure 11: Toy regression posterior density surfaces across different subspaces and sampling methods.