Improving Scalability and Inference in Probabilistic Deep Models

Simone Rossi

This dissertation is submitted for the degree of Doctor of Philosophy in the Doctoral School N. 130: Computer Science, Telecommunications and Electronics of Paris of the Sorbonne University

Committee in charge:

Maurizio Filippone	EURECOM	Advisor
Dino Sejdinovic	University of Oxford	Reviewer
Philipp Hennig	University of Tübingen	Reviewer
Maria A. Zuluaga	EURECOM	Examiner
Amandine Marrel	CEA-DES	Examiner
Marco Lorenzi	INRIA/Université Côte d'Azur	Examiner

Simone Rossi: *Improving Scalability and Inference in Probabilistic Deep Models,* © 21/02/2022

SUPERVISOR:
Maurizio Filippone

LOCATION:
Departement of Data Science, EURECOM (France)

TIME:
21/02/2022

To my family.

ABSTRACT

Throughout the last decade, deep learning has reached a sufficient level of maturity to become the preferred choice to solve machine learning-related problems or to aid decision making processes. At the same time, deep learning is generally not equipped with the ability to accurately quantify the uncertainty of its predictions, thus making these models less suitable for risk-critical applications. A possible solution to address this problem is to employ a Bayesian formulation; however, while this offers an elegant treatment, it is analytically intractable and it requires approximations. Despite the huge advancements in the last few years, there is still a long way to make these approaches widely applicable. In this thesis, we address some of the challenges for modern Bayesian deep learning, by proposing and studying solutions to improve scalability and inference of these models. The first part of the thesis is dedicated to deep models where inference is carried out using variational inference (VI). Specifically, we study the role of initialization of the variational parameters and we show how careful initialization strategies can make VI deliver good performance even in large scale models. In this part of the thesis we also study the over-regularization effect of the variational objective on over-parametrized models. To tackle this problem, we propose an novel parameterization based on the Walsh-Hadamard transform; not only this solves the over-regularization effect of VI but it also allows us to model non-factorized posteriors while keeping time and space complexity under control. The second part of the thesis is dedicated to a study on the role of priors. While being an essential building block of Bayes' rule, picking good priors for deep learning models is generally hard. For this reason, we propose two different strategies based (i) on the functional interpretation of neural networks and (ii) on a scalable procedure to perform model selection on the prior hyper-parameters, akin to maximization of the marginal likelihood. To conclude this part, we analyze a different kind of Bayesian model (Gaussian process) and we study the effect of placing a prior on all the hyper-parameters of these models, including the additional variables required by the inducing-point approximations. We also show how it is possible to infer free-form posteriors on these variables, which conventionally would have been otherwise point-estimated.

PREFACE

This thesis collects in a cohesive discussion several works published during the course of the PhD, which have been peer-reviewed by program committees in A* and A conferences and journals. Specifically,

- Simone Rossi, Pietro Michiardi and Maurizio Filippone. *Good Initializations of Variational Bayes for Deep Models*. In *International Conference on Machine Learning* (2019).
- Simone Rossi, Sebastien Marmin and Maurizio Filippone. *Walsh-Hadamard Variational Inference for Bayesian Deep Learning*. In *Advances on Neural Information Processing Systems* (2020)

  which extends the workshop paper:

  Simone Rossi, Sebastien Marmin and Maurizio Filippone. *Efficient Approximate Inference with Walsh-Hadamard Variational Inference*. In *Workshop on Bayesian Deep Learning* during the International Conference on Neural Information Processing Systems (2019).
- Simone Rossi, Markus Heinonen, Edwin V. Bonilla, Zheyang Shen, Maurizio Filippone. *Sparse Gaussian Processes Revisited: Bayesian Approaches to Inducing-Variable Approximations*. In *International Conference on Artificial Intelligence and Statistics* (2021)

In the last year of the PhD, I was involved in several collaborations with resulted in a series of papers that I co-authored

- Ba-Hien Tran, Simone Rossi, Dimitrios Milios and Maurizio Filippone. *All You Need is a Good Functional Prior for Bayesian Deep Learning*. Under review by the *Journal of Machine Learning Research*.

  which extends the workshop paper:

  Ba-Hien Tran, Dimitrios Milios, Simone Rossi, Maurizio Filippone. *Functional Priors for Bayesian Neural Networks through Wasserstein Distance Minimization to Gaussian Processes*. In *Symposium on Approximate Bayesian Inference* (2021)

- Ba-Hien Tran, Simone Rossi, Dimitrios Milios, Edwin V. Bonilla, Pietro Michiard, Maurizio Filippone. *Model selection for Bayesian Autoencoders* In *Advances of Neural Information Processing Systems* (2021)

Note that this thesis will contain only the parts of these works where my contributions are substantial. As a consequence, the chapter discussing these papers will partially refers to the original articles for a wider discussion and a more meaningful empirical evaluation.

Finally, for keeping the discussion coherent to the theme of this thesis, the manuscript will not include the following contribution

- Simone Rossi, Cristian Rusu, Lorenzo Rosasco and Maurizio Filippone. *Contributed discussion on "A Bayesian Conjugate Gradient Method" by J. Cockayne et al.* In *Bayesian Analysis 14(3)*

ACKNOWLEDGMENTS

In the past years, many great people have been part of my life and inspired and helped my work; I am pleased to finally have the opportunity to properly thank all those who contributed towards making this PhD such a remarkable experience.

First of all, I owe an immense gratitude to Maurizio Filippone for taking a huge gamble in a student in Electronic Engineering who wanted to do machine learning. While writing these acknowledgments, I went looking through the emails to find our first interaction. I was just asking Maurizio suggestions for some courses to take but we ended up discussing Gaussian processes and Bayesian statistics. From that meeting I remember the passion he had while explaining the research his team was doing (which I later found out it was just him and Kurt back then). A couple of years later, he convinced me to start this PhD, and I could never regret that decision. Thanks for being supportive and for guiding me in the immense literature of Bayesian inference; I would have been lost without his invaluable guidance. These were three years of many discussions, meetings and Skype calls and I will be forever in debt for always being on board with all the crazy ideas, including preparing a 4-hours tutorial on Bayesian deep learning in the middle of August! Overall he has inspired me to grow both as a researcher and as a person and I feel privileged to have worked under his supervision. Thanks also to the AXA Research Fund that, through the Chair of Computational Statistics awarded to Maurizio, has founded these three years.

Many thanks to Dino Sejdinovic and Philipp Hennig for accepting the invitation as reviewers of this manuscript, and thanks to Amandine Marrel, Maria A. Zuluaga and Marco Lorenzi for being in the examination committee.

Thanks to Pietro for the many inspiring chats we had in these years and for always providing encouragement, support and advice. I also owe Pietro a personal thanks for managing in his spare time a HPC cluster and for teaching me how to fully exploit it. Without this infrastructure and his lessons, many results and experiments would have not been possible.

Many thanks to the various collaborators, with whom I had the pleasure to work in these years. I am grateful to Sebastien Marmin and Dimitris Milios for dozen of useful discussions, not only related on the matter of this thesis. Thanks to Markus Heinonen for the discussion on point processes we had in Vancouver during NeurIPS and to Edwin V. Bonilla for his rigorous analyses and always insightful comments and questions. Finally, let me also acknowledge all the friends and fellow PhD students with whom I had the pleasure to share this journey; thanks to Kurt, Rosa, Gia-Lac, Jonas, Riccardo, Giulio, Ba-Hien, Bogdan, Davit, Graziano, for all the discussions and the amazing time spent together.

On a personal note, thanks to my family, and particularly my parents; without their constant love and support I would not be the person I am today. For your unwavering support and guidance, this thesis is dedicated to you.

CONTENTS

1 AN INTRODUCTION TO BAYESIAN DEEP LEARNING 1
1.1 A brief history of deep learning 2
1.2 Uncertainty quantification for decision making systems 4
1.2.1 Not all the uncertainties are the same 7
1.3 Bayesian deep learning 8
1.3.1 Modern inference for Bayesian neural networks 9
1.4 Today's challenges for Bayesian deep learning 11
1.4.1 Structure of this thesis 12

2 PROBABILISTIC METHODS FOR MACHINE LEARNING 15
2.1 An introduction to Bayesian machine learning 16
2.1.1 Bayesian model selection 17
2.2 Deep neural networks 18
2.2.1 Bayesian Neural Networks: Parameterization, Prior and Inference 20
2.3 Bayesian inference as optimization problem: Variational Inference . . 22
2.3.1 Optimization of the evidence lower bound (ELBO) 25
2.4 Sampling with scalable Markov Chain Monte Carlo 29

3 INITIALIZATIONS OF VARIATIONAL INFERENCE FOR BAYESIAN NEURAL NETWORKS 33
3.1 Overview 33
3.1.1 A review of the role of initialization in deep learning 36
3.2 Initialization of variational parameters: a proposed method 37
3.2.1 Initialization of DNNs for Regression 37
3.2.2 From the Bayesian linear model posterior to the variational approximation 39
3.2.3 Initialization for classification and convolutional layers 40
3.3 Experimental evaluation 42

3.3.1 The effect of initialization in deep variational neural networks 44
3.3.2 Scaling up variational inference to deep convolutional neural networks . . . 45
3.3.3 Comparison with variational inference beyond mean field Gaussian . . . 47
3.4 Final remarks . . . 50

4 EFFICIENT PARAMETERIZATIONS FOR VARIATIONAL POSTERIORS 53
4.1 The problem of overparameterization in variational inference . . . 53
4.1.1 Contributions . . . 55
4.2 Structured Approximations for Kernel Matrices . . . 56
4.3 From structured kernel approximations to Walsh-Hadamard Variational Inference . . . 57
4.3.1 Statistical properties of the structure induced by Walsh-Hadamard variational inference (WHVI) . . . 58
4.3.2 Reparameterizations in WHVI for stochastic optimization . . 62
4.4 Alternative structures, tensor factorization and extensions . . . 62
4.4.1 Extensions . . . 63
4.5 Empirical evaluation . . . 66
4.5.1 Toy example . . . 67
4.5.2 Empirical comparison on the UCI benchmark . . . 67
4.5.3 Bayesian convolutional neural networks for image classification 69
4.5.4 Comments on computational efficiency . . . 71
4.5.5 Exploring the parameter efficiency of WHVI with Gaussian processes . . . 73
4.6 Related work . . . 75
4.7 Final remarks . . . 76

5 THE EFFECT OF SELECTING THE PRIOR FOR BAYESIAN DEEP LEARNING 79
5.1 The choice of the prior matters . . . 79
5.1.1 Pathologies of deep prior functions . . . 81
5.1.2 Contributions . . . 82
5.2 The problem of choosing priors in the literature of Bayesian deep learning . . . 83
5.3 Imposing Gaussian process priors in Bayesian neural networks . . . 85

  5.3.1 A quick introduction to the Wasserstein distance 85
  5.3.2 Using the Wasserstein distance to impose the GP behaviour in BNN 87
  5.3.3 Prior Parameterization for Neural Networks 88
5.4 Empirical evaluation 90
  5.4.1 Visualization on a 1D regression synthetic dataset 92
  5.4.2 Comparison for Bayesian convolutional neural networks . . . 93
  5.4.3 Optimizing priors with data 94
5.5 Another route for Bayesian Occam's razor 96
  5.5.1 The distributionally-sliced Wasserstein distance 98
  5.5.2 Matching the marginal distribution to the data distribution via Wasserstein distance minimization 99
5.6 Model selection for Bayesian Autoencoders 100
  5.6.1 Formalization of Bayesian Autoencoders 101
  5.6.2 The pathology of standard priors for Bayesian auto-encoders (BAEs) and how to fix it 103
5.7 Concluding remarks 104

6 REVISITING THE APPROXIMATIONS FOR SCALABLE (DEEP) GAUSSIAN PROCESSES 107
6.1 Sparse Gaussian processes 107
6.2 Bayesian Sparse Gaussian Processes 110
  6.2.1 On scalable inference frameworks for GP models 111
  6.2.2 Sampling with VFE or FITC? 113
  6.2.3 Stochastic Updates Using the FITC Approximation 114
  6.2.4 An heteroskedastic version of the Gaussian likelihood 116
  6.2.5 Concluding Remarks 116
6.3 Practical considerations and extensions to deep GPs 117
  6.3.1 Prior choices 117
  6.3.2 Extension to deep Gaussian processes 119
6.4 Experiments 121
  6.4.1 Prior analysis and ablation study 122
  6.4.2 Choosing the objective: VFE vs FITC 124
  6.4.3 Deep Gaussian processes on UCI benchmarks 125
  6.4.4 Large scale classification 128
6.5 Concluding discussion 129

7 FINAL CONSIDERATIONS 131
7.1 Summary of the contributions and open problems 131
7.2 Is Bayesian deep learning solved? And now what? 134

BIBLIOGRAPHY 137

A ADDITIONAL DERIVATIONS 163
A.1 Recent advances on Bayesian inference for Deep Models 163
A.2 A primer of Wasserstein distance 164

B ADDITIONAL MATERIAL FOR CHAPTER 3 171
B.1 Experiments 171

LIST OF FIGURES

Figure 1.1 Evolution of accuracy and model complexity used for solving the ImageNet classification task 3
Figure 1.2 Autonomous driving agent learning to turn left while avoiding collisions, being being fast and smooth (image taken from the Tesla AI keynote 2021). 4
Figure 1.3 Visualization of point-estimated solutions and predictions with a probabilistic model. 6
Figure 1.4 Uncertainty decomposition using an heteroskedastic model of the *aleatoric uncertainty* and the *epistemic uncertainty* for the Motorcycle dataset. 7
Figure 2.1 Graphical representation of the generalization interpretation of the marginal likelihood 19
Figure 2.2 Example of a generic deep feedforward neural networks with L hidden layers. For visualization purposes the bias term is included in the notation $\boldsymbol{W}^{(l)}$. 19
Figure 2.3 Analysis of the distribution of samples from the prior for a shallow neural network with tanh activation. 20
Figure 2.4 Pictorial representation of running Bayesian inference on a simple architectures. The posterior and the predictive distributions are computed with numerical integration . . . 22
Figure 2.5 Pictorial illustration of the variational inference (VI) procedure, inspired by D. Blei et al. (2016). 23
Figure 2.6 Animation of a simple variational inference procedure, using analytical gradients and Monte Carlo estimation. 27
Figure 2.7 Graphical representation of the reparameterization trick. . . 29
Figure 3.1 Due to poor initialization VI fails to converge even after 1500+ epochs while with our iterative Bayesian linear modeling (IBLM) VI easily recovers the function after fewer iterations. 34

Figure 3.2 Visual representation of the proposed method for initialization. In (a) and (b), we learn two Bayesian linear models, whose outputs are used in (c) to infer the following layer. . . 38

Figure 3.3 Illustration of the different initialization for the variational parameters in a simple 1D regression task. 42

Figure 3.4 Progression of test mean negative loglikelihood (MNLL) with different initializations with shallow and deep architectures on Powerplant and Protein. Experiment repeated five times, only the average is shown. 44

Figure 3.5 Comparison of test MNLL after initialization of LENET for MNIST averaged out of eight successive runs. On the left, IBLM with different batch sizes, on the right comparison with Monte Carlo dropout (MCD). 46

Figure 3.6 On the **left**, comparison of initialization time versus test MNLL, averaged out of eight successive runs (on the **right**, magnification of the small portion of the plot). Orange corresponds to the Pareto frontier. Before training, four out of five optimal initializers are IBLM. 46

Figure 3.7 Progression of test error and test MNLL with different initializations on LeNet for MNIST and CIFAR10. Note that the Uniformative, Heuristic and Xavier intializers did not converge, and as such they are not included in the plots. . . 47

Figure 3.8 Progression of test error and MNLL for two different convolutional neural networks on both MNIST and CIFAR10. Variational inference is initialized with IBLM. 48

Figure 3.9 Reliability diagram and the expected calibration error for AlexNet trained on CIFAR10. 49

Figure 3.10 Progression of test error and MNLL for a very deep convolutional neural network trained on CIFAR10. 49

Figure 3.11 Test likelihood after initialization (on the left) and at the end of the training procedure (on the right) for LeNet on MNIST. 51

Figure 4.1 Normalized covariance of $\mathbf{g}$ and vect$(\boldsymbol{W})$. 57

Figure 4.2 Numerical verification of first and second moments for Walsh-Hadamard variational inference (WHVI) for one choice of $\mathbf{S}_1$, $\mathbf{S}_2$ and $q(\mathbf{g})$. The empirical quantities are obtained with Monte Carlo samples. 60

Figure 4.3 Ablation study of different structures for the parameterization of the weights distribution. 63

Figure 4.4 Comparison between Hadamard factorization in WHVI and tensor factorization. The number in the parenthesis is the hidden dimension. Plot is w.r.t. iterations rather then time to avoid implementation artifacts. The dataset used is Drive. 65

Figure 4.5 Regression example trained using WHVI with Gaussian vector (1541 param.) and with planar normalizing flow (10 flows for a total of 4141 param.), mean-field Gussian (35k param.) and Monte Carlo dropout (MCD) (17k param.). . . . 67

Figure 4.6 Comparison of the test MNLL as a function of the number of hidden units. The dataset used is Powerplant. 69

Figure 4.7 Comparison of the test MNLL as a function of the number of hidden units and hidden layers. 70

Figure 4.8 Analysis of the training curve (on the left) and test curve (on the right) for the Mocap dataset. 71

Figure 4.9 Reliability diagram and expected calibration error of VGG , ALEXNET and RESNET with WHVI. 72

Figure 4.10 Test curve of LeNet when trained with different combination of variational inference parameterization. 73

Figure 4.11 Analysis of the timing efficiency of WHVI. **(a)** Inference time on the test set with 128 batch size and 64 Monte Carlo samples. Experiment repeated 100 times on an GPU fully dedicated to it. **(b)** Timing comparison of different implementations of the vectorized Fast Walsh-Hadamard transform, with a batch size of 512 data points. 74

Figure 4.12 Comparison of the test MNLL as a function of the memory footprint for four datasets. 74

Figure 4.13 Power profiling during inference on the test set of CIFAR10 with ALEXNET The task is repeated 16 consecutive times and profiling is carried out using the `nvidia-smi` tool. . . . 75

Figure 4.14 Analysis of the parameter efficiency of WHVI for scalable Gaussian process regression with variational inference. . . . 75

Figure 5.1 Test performance on CIFAR10 with different priors. Figure replicated from the arXiv version of A. G. Wilson and Izmailov (2020a). 80

Figure 5.2 Grid search analysis on the prior variance for LeNet trained via variational inference on CIFAR10. 81

Figure 5.3 Prior functions of a fully-connected Bayesian neural network (BNN) with 2, 4 and 8 layers obtained by placing a Gaussian prior on the weights and using a Gaussian process (GP) with two different kernels. 83

Figure 5.4 Schematic representation of the process of imposing GP priors on BNNs via Wasserstein distance minimization. . . . 88

Figure 5.5 Visualization of one-dimensional regression example with a three hidden-layer multilayer perceptron (MLP). The first two rows illustrate the prior sample and distributions, whereas the last two rows show the corresponding posterior distributions. The means and the 95% credible intervals are represented by red lines and shaded areas, respectively. The middle row shows progressions of the prior optimization. . 92

Figure 5.6 A timing comparison between imposing functional prior and cross-validation with grid-search and using Bayesian optimization. 95

Figure 5.7 Different loss functions for model selection share the same optimum, for linear models. 97

Figure 5.8 Realizations sampled from different priors given an input image. OOD stands for out-of-distribution. 102

Figure 6.1 Representation of the induced distribution on the covariance function at location x when placing priors on different set of parameters. 109

Figure 6.2 Illustration of a binary classification task on the BANANA dataset. *Left*: the decision bounds of the average classifier. *Right*: the posterior marginals of the inducing inputs. 118

Figure 6.3 Visual representation of a 2-layer deep Gaussian process (DGP). 119

Figure 6.4 Analysis of different priors on inducing locations for Bayesian Sparse Gaussian Process (BSGP) on the UCI benchmark datasets for determinantal point process (DPP), Strauss process, uniform and normal priors on $\mathbf{Z}$. 121

Figure 6.5 Ablation study on the effect of performing posterior inference on different sets of variables. From SVGP, where the posterior is constrained to be Gaussian and the remaining parameters are point-estimated, to our proposal BSGP, where we infer a free-form posterior for all $\Psi = \{\mathbf{u}, \theta, \mathbf{Z}\}$. We refer the reader to Table 6.1 for details on the methods (colors are matched). 122
Figure 6.6 Analysis of different choices of objectives when used for optimization and sampling. We refer the reader to Table 6.1 for a description of the methods. 123
Figure 6.7 p-values of the hypothesis test that BSGP with variational free energy (VFE) objective is better than BSGP with fully independent training conditional (FITC) objective; depth of the DGP from 1 to 5. For models with p-values < 0.05, we reject the hypothesis 123
Figure 6.8 BSGP in with different depths of the DGP with two different objective: FITC and VFE. The number of layers corresponds to the depth of the DGP. 124
Figure 6.9 Test MNLL on UCI regression benchmarks (the error bars represent the 95%CI). The lower MNLL (i.e. to the left), the better. The number on the right of the method's name refers to the depth of the DGP. *Bottom right*: Rank summary of all methods. 125
Figure 6.10 Comparison of test MNLL as function of training time. The dashed line on the right hand side plot corresponds to SVGP with $M = 1000$ inducing points. 126
Figure 6.11 Comparison of test MNLL as a function of prediction time on the largest dataset (Protein). 127
Figure 6.12 Comparison with structured inducing variables methods. KISS-GP could only run on the Powerplant dataset (hence the ✗ on Protein and Kin8NM). 128
Figure 7.1 Number of unique new papers submitted on arxiv with title or abstract containing different permutations of *Bayesian deep learning*. Data crawled from the public API on December 8th 2021. 134
Figure b.1 Progression of test RMSEand test MNLL with different initializations on a shallow architecture. 172

Figure b.2 Progression of test RMSEand test MNLL with different initializations on a deep architecture. 173
Figure b.3 Progression of test RMSEand test MNLL with different initializations on a deep architecture. 173

LIST OF TABLES

Table 4.1 Time and space complexity of various approaches to variational inference (VI). 64
Table 4.2 Test root mean squared error (RMSE) and test mean negative loglikelihood (MNLL) for regression datasets. Results in the format "*mean (std)*" 68
Table 4.3 Test performance of different Bayesian convolutional neural networks (CNNS). 72
Table 5.1 Glossary of methods used in the experimental campaign. . . 91
Table 5.2 Results for different convolutional neural networks on CIFAR10. Experimental comparison performed in B.-H. Tran et al. (2020) by the first author (Ba-Hien Tran). 94
Table 6.1 A summary of previous works on inference methods for Gaussian processs (GPS). θ, u, Z refer to the GP hyperparameters, inducing variables and inducing inputs, respectively. (✗) indicates that variables are optimized. 108
Table 6.2 AIRLINE dataset predictive test performance. 129
Table 6.3 HIGGS dataset predictive test performance. 129

1

AN INTRODUCTION TO BAYESIAN DEEP LEARNING

The *Deep Learning revolution* has arguably reshaped the way we do machine learning, both from a methodological and from an application point of view (LeCun, Bengio, et al., 2015). Throughout the last decade, the practical advancements and the theoretical understanding of deep learning models and practices has arguably reached a level of maturity such that it is the preferred choice for any practitioner seeking simple yet powerful solutions to solve machine learning-related problems. In few years the community made tremendous advances to develop better and more efficient learning algorithms and architectures. Combined with massive availability of data and implementations that can easily scale from small edge devises to huge data-center of cloud-serviced clusters of graphic processing units (GPUs), deep learning quickly has gained a place in our society, from personal assistants (Su et al., 2021; Muralidharan et al., 2019), autonomous driving cars (Kendall, Hawke, et al., 2019; Kiran et al., 2021; Yurtsever et al., 2020; Grigorescu et al., 2020), automated medical diagnostics (Dang et al., 2022; Chan et al., 2020; Wu, 2021) and drug synthesis (Jumper et al., 2021; Jiménez-Luna et al., 2020). At the same time, this level of dissemination raises questions on how much we blindly rely on these model's predictions, or how fair they are (Guo et al., 2017; Du et al., 2021). Taking informed decisions is the key to build reliable and trustworthy systems and it is only possible with a sound modeling of the uncertainty, both in the data and in the model. This is even more critical for applications for which accuracy is not the only important performance metric and for which having highly calibrated confidence in the predictions is a strict system requirement.

1.1 A BRIEF HISTORY OF DEEP LEARNING

In an attempt at providing an historical review of neural networks, Schmidhuber (2015) traces back their origins to early XIX century (Legendre, 1805; Gauss, 1809; Gauss, 1821), arguing that deep neural networks are in practice variants of linear regression methods. For what we consider nowadays neural networks we need to wait the 1950's, when early success stories involve an attempt at solving the learning problem (Hebb, 1949) and the formalization of the perceptron (Rosenblatt, 1958; Rosenblatt, 1962). Arguably the first important milestone for deep learning is *backpropagation* (also known as *reverse mode differentation*), an efficient procedure for error minimization through the combination of (stochastic) gradient descent (Hadamard, 1908; Robbins and Monro, 1951) and the chain rule of differentiation (Leibniz, 1676). This was preliminarily discussed by Linnainmaa (1970) but applied to learning of neural networks later by Rumelhart et al. (1986). This simple method was extended in the following years to speed-up convergence (Vogl et al., 1988; Battiti, 1989), by e. g. using second order derivative information through the diagonalization of the Hessian (Becker and LeCun, 1989; P. Simard and LeCun, 1993), by adapting the global step size (LeCun, P. Y. Simard, et al., 1993; X.-H. Yu et al., 1995) or by computing individual learning rates for each parameter (Jacobs, 1988; Silva and Almeida, 1990). In more recent years, these methods have become the backbone of optimizers like ADAGRAD (Duchi et al., 2011), ADADELTA (Zeiler, 2012), RMSPROP (Schaul et al., 2013) and ADAM (Kingma and Ba, 2015). Having now the tool for learning (i. e.optimization), research moved to study new architectures and exploit new computational devices, like GPUs. Convolutional neural networks (CNNs), for example, were introduced for improving the performance of deep networks on highly structured data (like images for classification tasks) (LeCun, Boser, et al., 1989; LeCun, Boser, et al., 1990; LeCun, Bottou, et al., 1998) while long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), a more powerful class of recurrent neural networks (RNNs), were found to learn long term dependences in the data, which was previously impossible to do (Pérez-Ortiz et al., 2003). By early 2000's after initial implementations of neural networks on GPUs were proved successful with speed-up factors up to 20 times (Chellapilla et al., 2006; Oh and Jung, 2004), it was clear that the next milestone would come by solving the computational issues of these models and by providing open source implementations for heterogeneous computing devices. After effectively solving the classification problem of handwritten digits on MNIST (Ranzato et al., 2006; Ciresan et al., 2012), Krizhevsky, Sutskever, et al. (2012) proposed to use a multi-GPU implementation of a deep CNN (later known as ALEXNET) and with

For an up-to-date discussion of deep learning optimizers, the recent work by Schmidt et al. (2021) benchmarks fifteen popular optimizers in eight different problems, for a total of 53 760 *experiments (after hyper-parameters tuning).*

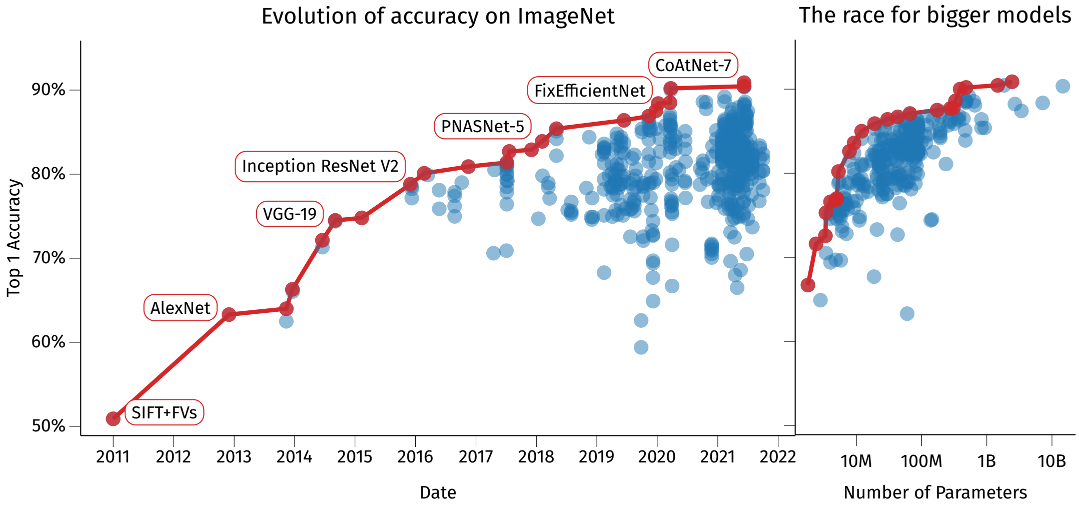


Figure 1.1: Evolution of accuracy and model complexity used for solving the ImageNet classification task. Data extracted in September 2021 from `https://paperswithcode.com/sota/image-classification-on-imagenet`

this architecture they achieved the best result on IMAGENET, a popular computer vision benchmark. This was the first time that a deep learning approach was able to outperform manual feature extraction and classic computer vision algorithms. Since then, accuracy on this benchmark is continuously increasing, as well as the size of the models implemented (see Figure 1.1). The availability of libraries with automatic differentation (Baydin et al., 2017) and highly efficient implementations, like Tensorflow (Abadi et al., 2015), PyTorch (Paszke et al., 2019) and JAX (Bradbury et al., 2018), made deep learning the preferred choice for any practitioner seeking simple yet powerful solutions to solve machine learning related problems. Examples of such wide adoption of deep learning come from cosmology (Dieleman et al., 2015; Lanusse et al., 2019; Fussell and Moews, 2019), experimental physics (Acciarri et al., 2017; Aurisano et al., 2016) and neuroscience (J. X. Wang et al., 2018; Yamins and DiCarlo, 2016; Bellec et al., 2018; Zador, 2019), but it has cross-fertilized other computer science fields, such as digital hardware design (Kwon and Carloni, 2020; F. Zhang et al., 2020; Y. C. Lu et al., 2020), data management systems (Kraska et al., 2018; G. Li et al., 2019; H. Liu et al., 2015; Marcus and Papaemmanouil, 2019) and materials science (Zhizhou Zhang, 2021; Kim et al., 2020; Z. Zhang and Gu, 2020; Wilt et al., 2020).

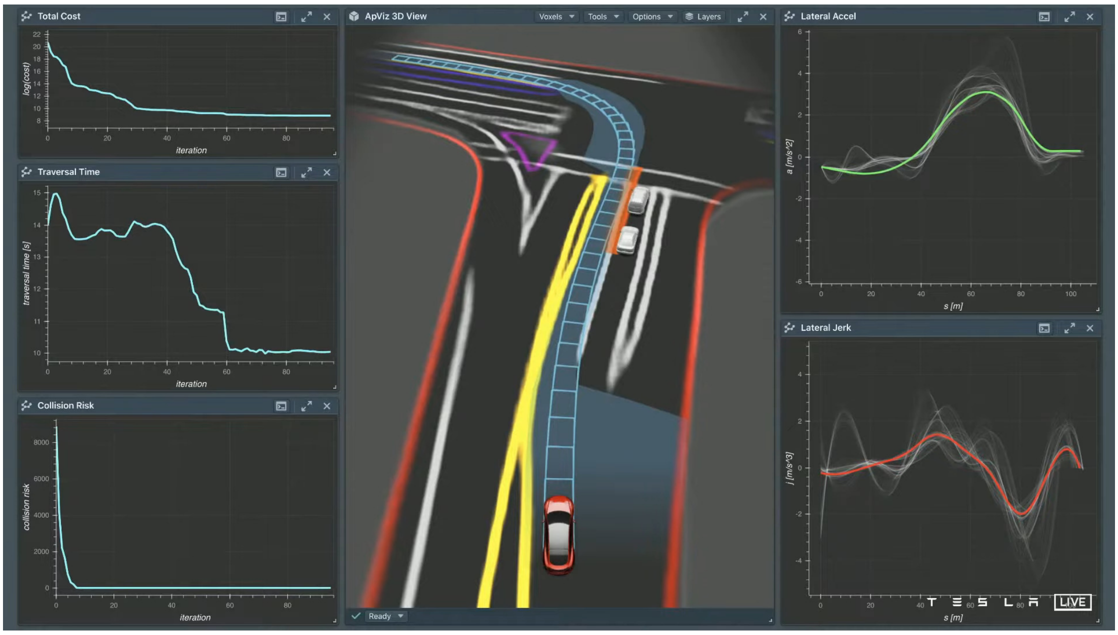

Total Cost
log(cost)
iteration
Traversal Time
traversal time [s]
iteration
Collision Risk
collision risk
iteration
ApViz 3D View
Voxels
Tools
Options
Layers
Ready
Lateral Accel
a [m/s^2]
s [m]
Lateral Jerk
j [m/s^3]
s [m]
TESLA
LIVE

Figure 1.2: Autonomous driving agent learning to turn left while avoiding collisions, being being fast and smooth (image taken from the Tesla AI keynote 2021).

1.2 UNCERTAINTY QUANTIFICATION FOR DECISION MAKING SYSTEMS

Deep learning models, like any other machine learning tools, are generally only a part of larger *systems* with *agents* interacting in complex *environments*. Take as a simple example the problem of autonomous driving (Figure 1.2): the system collects data from multiple sensors (video, Lidar, etc.), it controls the car by e.g. accelerating, braking or turning the steering wheel and it interacts with a chaotic environment made of other cars, obstacles and pedestrians. The output of a model can be used for taking *actions*, like performing an emergency breaking if another car suddenly changes lane. Generally, decision making systems combine beliefs and data to optimize an *utility function*. For example, in Figure 1.2 the agent is learning to turn left while minimizing jointly the risk of collision, the time required to complete the manoeuvre and the lateral acceleration and jerk. Unfortunately, real systems are hindered by uncertainty, whether due to partial observability, non-determinism, or a combination of the two. This means that for example we may never know *for certain* what state the system would be in after taking a sequence of actions, as predicted by our model. As a consequence, utility theory needs to take into account the unidentifiability of the problem by means of probability theory, which can express the degree of belief an agent has of the model and the environment.

Going back to the example above, it can happen that the other car is actually not changing lane but a different shadow is casted on the road in such a way that it looks like.

Quantification of the uncertainty becomes then essential for taking informed decisions. Deep learning models, while being very flexible and powerful models, they are usually not equipped with the ability to accurately estimate the uncertainty of their predictions. Learning algorithms for these models were typically designed through the lens of statistical learning theory by means of minimizing the expected risk defined as

$$\mathcal{R}(f) = \int e(f(\mathbf{x}), \mathbf{y}) p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \tag{1.1}$$

where $f(\cdot)$ defines a neural network function in the space of hypothesis $\mathcal{F}$ and $e(\cdot, \cdot)$ is a properly defined error function. Due to the intractability of this formulation, in practice we rely on the empirical risk as proxy to the expected risk and it formulates the learning problem as

$$\min_{f \in \mathcal{F}} \widetilde{\mathcal{R}}(f) = \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} e(f(\mathbf{x}_i), \mathbf{y}_i) \tag{1.2}$$

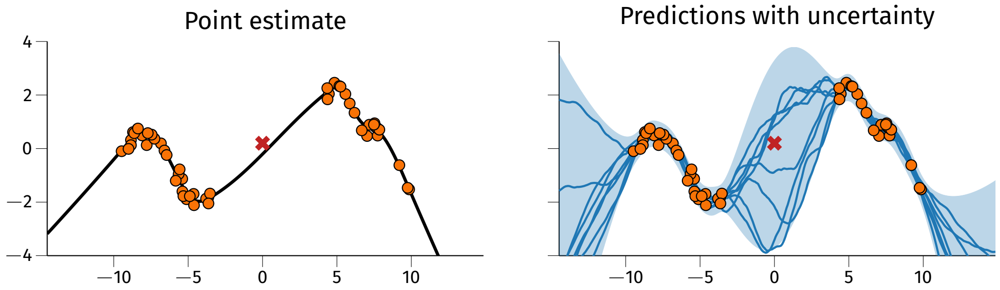
where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the training data. Stability analysis of the learning algorithm with respect to changes in the training data suggest that in order to build models with good generalization properties, we need to impose regularization which controls the tradeoff between data fitting and model variance. For example, a generic regularization algorithm (e.g. Tikhonov regularizer) minimizes the empirical risk plus a stabilizer on the model parameters,

$$f_n = \arg\min_{f \in \mathcal{F}} \left(\widetilde{\mathcal{R}}(f) + \lambda \|f\|_{\mathcal{F}} \right) \tag{1.3}$$

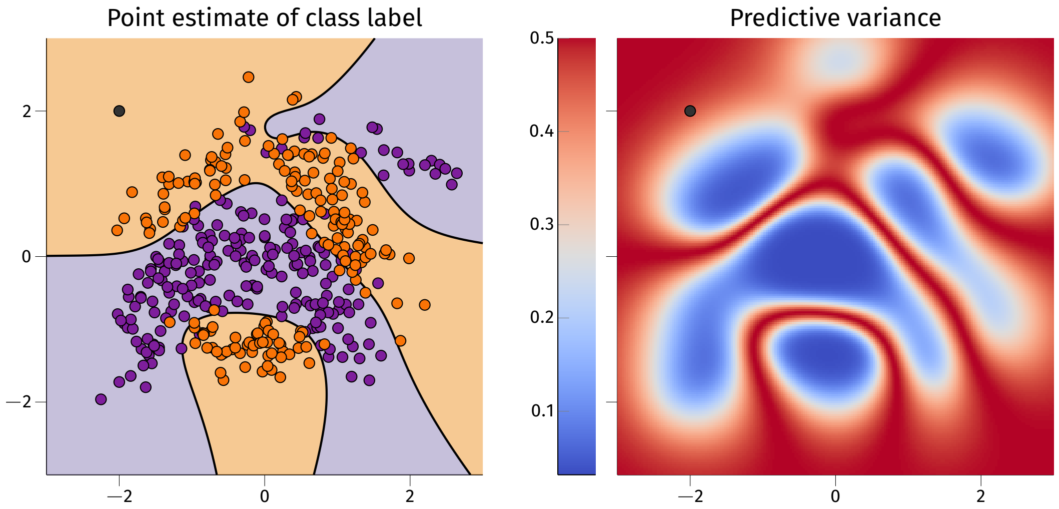
By building a learning algorithm is this way we obtain a solution f_n, which under some conditions, exhibit a bounded generalization error:

$$\mathbb{P}\left(\left| \widetilde{\mathcal{R}}(f_n) - \mathcal{R}(f_n) \right| \geqslant \varepsilon \right) \leqslant \mathbb{P}\left(\sup_{f} \left| \widetilde{\mathcal{R}}(f) - \mathcal{R}(f) \right| \geqslant \varepsilon \right) \tag{1.4}$$

Note that in practice this way of designing learning algorithms implicitly aims at finding *the best solution*, which equivalently minimizes some expected generalization error. Regression models return a single vector of predictions, while for



(a) Regression with noisy data.



(b) Binary classification in two dimensions.

Figure 1.3: Visualization of point-estimated solutions and predictions with a probabilistic model.

classification the output represents a probability vector for the associated set of classes.

One way to reason about the confidence in the model's predictions is by treating the problem probabilistically. In practice, instead of having predictions which are point-estimated, we can rely on proper probability distributions around the model outputs. In this way, we can reason about the confidence in the predictions of the model and quantify the risk associated for a down-stream decision making process. Bayesian machine learning offers a powerful and intuitive way to propagate belief using data. Figure 1.3 visually represents two machine learning tasks (regression and binary classification) which are learned with regularized loss (on the left) and probabilistically with Bayesian inference (on the right). Note how the additional information about prediction confidence at a certain test point can be used in down-stream tasks to build more reliable decision making systems.

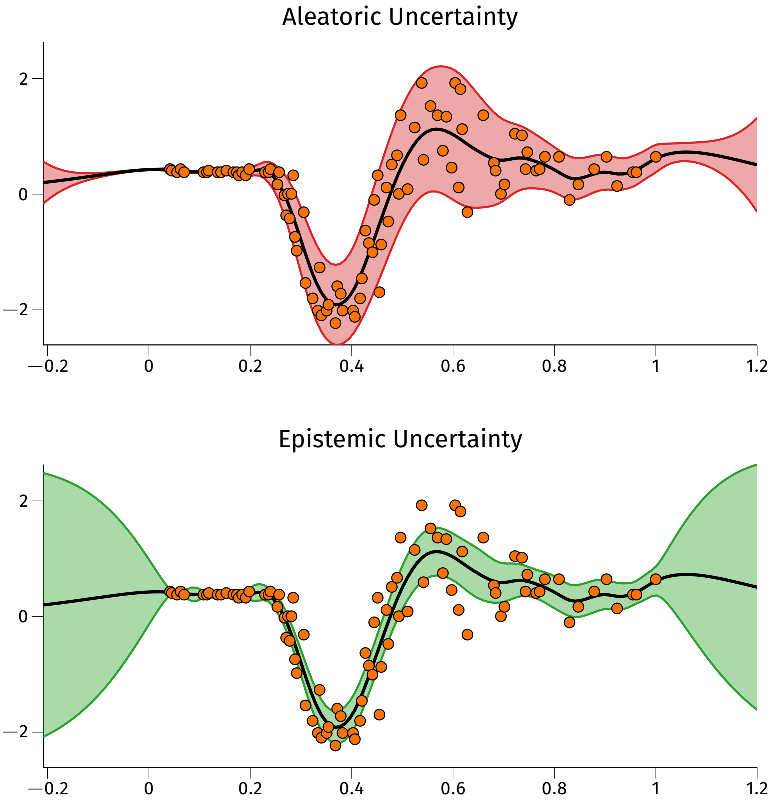


Figure 1.4: Uncertainty decomposition using an heteroskedastic model of the *aleatoric uncertainty* and the *epistemic uncertainty* for the Motorcycle dataset.

1.2.1 Not all the uncertainties are the same

Before diving into more details of Bayesian inference and how it can be implemented in deep learning models, we need to briefly discuss the role of uncertainty in practice. Broadly speaking, the sources of uncertainty we can distinguish are twofold: we are either uncertain of the data or we are uncertain of the model and its parameters. The former, also known as *aleatoric uncertainty*, can be due to several factors, including for example noisy data acquisition systems. This uncertainty is *irreducible*, in the sense that in general we have no controls over it and it's independent on the amount of data available. The second kind of uncertainty is related to the model parameters and it is usually referred to as *epistemic uncertainty*. This uncertainty is due to the fact that there are up to an infinite amount of possible models that are able to explain a given dataset. As a consequence, we could be unsure about which model parameters we need to choose to predict with. Differently from the aleatoric uncertainty, the epistemic is in fact *reducible*: once we collect enough data to estimate accurately the model's parameters such uncertainty decreases (even down to zero, in the infinite data limit). On the other hand, by tautology for regions away from the data the model has no information and it will exhibit higher uncertainty. Note that this could be useful to detect *out of distribution* test data, i.e. situations for which the model is inferring on input points that are far away from training data.

In Figure 1.4, we have a simple example where the aleatoric and the epistemic uncertainties are decomposed and visualized. Note, for example, how the epistemic uncertainty behaves far away from the training data, where due to missing information the variance is larger than in the region with data.

1.3 BAYESIAN DEEP LEARNING

In this section, we briefly review some relevant literature for Bayesian neural networks, from their inception to more recent developments. First examples of Bayesian neural networks can be traces back to late '80s. Researchers at the AT&T Bell Laboratories involved in the general problem of learning neural networks from examples discussed the possibility of using Bayesian inference for achieving this goal. J. Denker et al. (1987) proposes to place an uniform prior distribution on the space of the weights and it also shows how this translates into the function space, effectively by building a network for each configuration of the weights. Few years later Tishby et al. (1989) derives a theoretical framework for applying Bayes rule to layered models by assigning a likelihood, a nonsingular prior on the "configuration space" (i.e. the weight space) and by deriving the posterior as a Gibbs canonical distributions on the ensemble of networks. As Yarin Gal pointed out in his thesis (Gal, 2016), this is possibly the first example of modern Bayesian neural network (BNN) formalized by means of Bayes rule. The framework of Tishby et al. (1989) was put in practice by J. S. Denker and LeCun (1990), who propose to use the recently developed back-propagation algorithm for the Laplace approximation of the posterior. D. J. C. MacKay (1991) discusses the problem of Bayesian model selection and the effect of the Occam factor for the choosing the parameters of a neural network. Using the Laplace approximation he also shows how the model evidence correlates with the test error and generalization. D. J. C. MacKay (1992) extends some of these results, showing how the model evidence can be used to select the number of hidden units. Approximations for Bayesian neural networks were not limited to the Laplace method only. In the same period, Neal (1992a) and Neal (1992b) discuss how Markov chain Monte Carlo

(MCMC) can be efficiently implemented in Bayesian neural networks, specifically by using the Hamiltonian Monte Carlo (HMC) framework introduced by Duane et al. (1987). In a follow-up work, Neal (1994b) analyzes the infinite limit behaviour of Bayesian neural networks priors and he draws important connections with another Bayesian model, the Gaussian process (GP). All these results are collected in his thesis (Neal, 1996; Neal, 1994a). Variational inference (VI) has been also proposed as a scalable approximation for these models. Barber and C. Bishop (1998) derives an objective which forms a lower bound of the marginal likelihood. They proposes to use full covariance matrices (differently from what previously done by G. E. Hinton and Camp, 1993) and a Gamma hyper-prior on the model parameters and they derive a procedure to perform tractable VI with free-form variational posteriors.

1.3.1 Modern inference for Bayesian neural networks

More recently, Graves (2011) proposes a practical way to perform variational inference in a scalable way on neural networks. He discusses several important aspects of the VI framework, like the role of the posterior approximation and how it relates to regularized loss training, how the prior parameters can be adapted during training and finally how Monte-Carlo sampling can be used to approximated the intractable expectation of the log-likelihood. This work was later extended by Blundell et al. (2015) with the so-called *Bayes by backprop*, which shows how it is possible to use the reparameterization trick to generate unbiased estimates of the variational parameters' gradients. Alternatives can be found in the *probabilistic backpropagation* (Hernandez-Lobato and R. Adams, 2015), in *Stein variational inference* (Q. Liu and D. Wang, 2016) and in the *Monte Carlo dropout* (Gal and Ghahramani, 2016b). The latter found its fortune (Kendall and Gal, 2017) in the embarrassingly simple implementation required to compute the output multiple times with different dropout masks, with nonetheless drawing important theoretical connections with variational inference (Y. Li and Gal, 2017).

Research in VI for BNN has been also driven by the necessity of moving away from the simple fully factorized Gaussian assumption for the posterior. In this line of works, we can find attempts at parametrizing efficiently *matrix variate posteriors* (Louizos and Welling, 2016; Sun, C. Chen, et al., 2017; G. Zhang et al., 2018; Rossi, Marmin, et al., 2020). Note that only around this time we start to see more complex neural network architectures (e.g. CNNs) being employed in the variational setting. Osawa et al. (2019)—building on top of the work of M. E. Khan, Nielsen, et al. (2018)—shows how variational inference with natural gradients can scale to ResNet-18 trained on ImageNet. Changing the perspective of VI, Sun, G. Zhang, et al. (2019) proposes to perform variational inference in function space, by defining and optimizing the lower bound directly on the stochastic processes (functional output). Due to the specific kind of priors used, Sun, G. Zhang, et al. (2019) only limit their analysis to regression tasks, leaving classification and other problems as a future work. In a recent work by Farquhar et al. (2020), the Authors challenge the assumption that structured posteriors are better than the mean-field approximation. In essence, among other things, they speculate that for a given BNN with full covariance there exists a deeper BNN which can induce the same functional posterior in function-space.

As an alternative to approximating the posterior with variational inference, it is also possible to use other kind of approximations. With the Laplace approximation (LA), for example, one would choose to fit a Gaussian distribution around the (local) maximum of the joint likelihood (D. J. C. MacKay, 1998; Ritter et al., 2018; Immer, Bauer, et al., 2021). Recently this approximation has found new life also thanks to efficient implementations for computing the Hessian of neural networks (Dangel et al., 2020; E. A. Daxberger et al., 2021). Immer, Korzepa, et al. (2021), for example, shows how we can build a locally linearized model starting from the Laplace approximation, while Kristiadi et al. (2020) discusses how the LA at the last layer is enough to fix the issue of overconfidence in neural networks.

Instead of approximating the posterior, we could also generate samples from this intractable distribution using some gradient-based MCMC. The main limitation of such approaches reside in the expensive computations required to evaluate the gradients of the logarithm of the unnormalized posterior density in big data regimes (Neal, 2011). Unfortunately, as Betancourt (2015) discusses and quantifies, mini-batching the data fundamentally breaks and compromises the scalability and accuracy of näive HMC. To solve this problem, T. Chen et al. (2014) proposes to add an additional friction term to counter-balance the effect of the noise introduced by subsampling the data. Notably, T. Chen et al. (2014) draws significant connections between the proposed Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) and the classic stochastic gradient descent (SGD), which has been leveraged by Springenberg et al. (2016) to estimate some of the hyper-parameters HMC requires. To improve the performance and the exploration efficiency of SGHMC, R. Zhang et al. (2020) proposes to cycle through different values of step sizes while Franzese et al. (2021a) relaxes the assumption of unisotropic gradient noise. With a careful choices of the hyper-parameters, stochastic gradient HMC methods are shown in theory to converge to the exact posterior (Franzese et al., 2021b) and in practice to deliver high quality samples compared with classic HMC (Izmailov et al., 2021).

Finally, ensembles of models could be employed as Bayesian approximation (Newton and Raftery, 1994; Garipov et al., 2018). Lakshminarayanan et al. (2017) proposes *deep ensembles* as a simple way to train a set of independent models and combine their predictions to generate informative uncertainty estimation. In a more recent work, B. He et al. (2020) generalize this setup to infinitely wide neural networks using the neural tangent kernel (NTK). As an attempt to theoretically characterize the learning behavior of these methods, Milios, Michiardi, et al. (2020) analyzes the problem, finding that training these models is equivalent to a Kullback-Leibler (KL) minimization akin to variational inference.

1.4 TODAY'S CHALLENGES FOR BAYESIAN DEEP LEARNING

Despite the fact that we have witnessed huge advancements in the last few years, arguably Bayesian deep learning is still in its infancy, especially from a practical point of view. Broadly speaking, we can summarize the current challenges in two categories, which are tightly entangled: *inference* and *scalability*. Bayesian inference, while offering an elegant formulation, cannot be solved exactly. The approximations required could decrease the quality of the predictions while certainly increasing the complexity of the entire machine learning pipeline. Variational inference, for example, increases the model parameters to be optimized by at least a factor of two, while sampling methods typically require orders of magnitude more gradients evaluations. Also, they require to store the samples of the parameters' posterior, which increases the memory footprint of these algorithms hundredfold. On top of this, we are implicitly augmenting the challenges by adding the problem of choosing the prior distribution. This fundamental building block of the Bayesian theory is proved empirically to be quite critical and yet difficult to assess its quality *a-priori*. On the other hand, we can fortunately find comfort from analyzing other Bayesian models, like GPs and deep Gaussian processs (DGPs), which are closely related to Bayesian neural networks.

1.4.1 Structure of this thesis

In this thesis, we will attempt at analyzing some of the aforementioned limitations and proposing some solutions. The rest of the thesis is structured as follows:

- Chapter 2 is dedicated to a quick review of Bayesian machine learning and to an introduction to approximations to intractable inference for Bayesian neural networks.

- Chapter 3 will discuss the challenges of initializing variational inference for deep neural networks. We found that this problem is more severe for complex architectures, such as deep convolutional neural networks, for which VI systematically converges to trivial solutions (posterior equal to the prior). We also propose a novel initialization strategy for VI, whose performance will be assessed in various experiments.

- Chapter 4 proposes a new structured parameterization for variational inference, inspired by the wide literature on scalable kernel machines. The key operation within our proposal is the Walsh-Hadamard transform, and this is why we name our proposal Walsh-Hadamard variational inference (WHVI). Unlike mean field VI, WHVI induces a matrix-variate distribution to approximate the posterior over the weights, thus increasing flexibility at a log-linear cost in D. We derive expressions for the reparameterization and the local reparameterization tricks, showing that, the computational complexity is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$.

- Chapter 5 shows how important is the role of the prior for improving the performance of Bayesian deep learning models and it will address the challenges to accurately selecting them. In more details, we seek to tune the prior distributions over BNNs parameters so that the induced functional priors exhibit interpretable properties, similar to shallow GPs. While BNN priors induce a regularization effect that penalizes large values for the network weights, a GP-adjusted prior induces regularization directly on the space of functions. We will show how we can use the *Wasserstein distance* between the distribution of BNN functions induced by a prior over their parameters, and a target GP prior.

- Chapter 6 will revisit model approximations and inference schemes for scalable Gaussian processes and deep Gaussian processes. We will discuss the role of the inducing inputs in GP models and their treatment as variational parameters or even hyper-parameters. Given their potential high dimensionality and that the typical number of inducing variables goes beyond hundreds/thousands, we argue that they should be treated simply as model variables and, therefore, having priors and carrying out efficient posterior inference over them is an important—although challenging—problem.

- Chapter 7 will finally present some concluding remarks and analyze some possible future trends.

2

PROBABILISTIC METHODS FOR MACHINE LEARNING

Starting in 1762, Richard Price collected and rewrote some of the notes of his friend Reverend Thomas Bayes, who tragically pasted away the year before in 1761. In 1763, Price presented his friend's work at the *Royal Society* and in 1764 he published an edited version in the *Philosophical Transactions of the Royal Society of London*. In "An Essay towards solving a Problem in the Doctrine of Chances" Bayes tries to solve the following inferential problem:

> Given the number of times in which an unknown event has happened and failed. [It is required to calculate] the chance that the probability of its happening in a single trial lies somewhere between any two degrees of probability that can be named.
>
> — Thomas Bayes (1763)

In his notes, Bayes derives several theorems that will constitute the backbone of modern probability theory, with particular emphasis on conditional probabilities. He then proposes an argument for using a uniform distribution for the binomial parameter $p(0 \leqslant \pi \leqslant 1)$ to represent absence of knowledge about it (Edwards, 1978). In practice, this is the formulation of what we refer to as *Bayes' rule* or *Bayes' theorem*, as a way to solve the inverse conditional probability problem. This new change of prospective to modeling reality is arguably one of the most important milestone in history of Science and Mathematics. Among others, Sir Harold Jeffreys prizes the accomplishments of Bayes by recognizing Bayes' theorem to be "to the theory of probability what the Pythagorean theorem is to geometry" (Jeffreys, 1939).

This chapter builds the foundations upon which the rest of the thesis will lie. Specifically, we start in § 2.1 with a review of basis concept of probability theory and the use of Bayes' rule in machine learning. To this goal, we present some important results of Bayesian inference approximations, notably variational inference (VI) and Markov chain Monte Carlo (MCMC) sampling.

2.1 AN INTRODUCTION TO BAYESIAN MACHINE LEARNING

Bayes offers us a principled machinery to do probabilistic modeling: the way of reasoning about uncertainty, a structured procedure to include prior knowledge of the problem (or the lack of thereof) and a formal methodology to model selection are only some of the characteristics that makes Bayes' theorem and Bayesian inference very well suited for machine learning (Ghahramani, 2013; Ghahramani, 2015).

In case of supervised learning, we assume both $\mathbf{X}$ *and* $\mathbf{y}$ *to be known. The Bayesian framework can also be applied to unsupervised tasks and generative modeling.*

Let's dive a bit more on the theory of Bayesian inference. Assume a generic parametric model f parameterized by some unknown parameters θ (i.e. $f(\cdot, \theta)$) and a collection of data $\mathbf{y} \in \mathbb{R}^N$ corresponding to some input points $\mathbf{X} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^{D_{\text{in}}}\}_{i=1,\dots,N}$. A generic formulation of the Bayes' rule takes the following form,

$$\overbrace{p(\theta \mid \mathbf{y})}^{\text{Posterior}} = \frac{\overbrace{p(\mathbf{y} \mid f(\mathbf{X}, \theta))}^{\text{Likelihood}} \; \overbrace{p(\theta)}^{\text{Prior}}}{\underbrace{p(\mathbf{y})}_{\text{Model Evidence}}}. \tag{2.1}$$

In order to keep the notation uncluttered, in the following chapters we might avoid to explicitly write the dependency on $\mathbf{X}$; nonetheless, f is always computed at some input points $\mathbf{x}$. Note also that everything remains valid even if the output dimensionality is greater then one. Depending on the context, $\mathbf{y}$ could be a vector or a matrix $N \times D_{\text{out}}$. Each part of this equation has a specific role. The *prior* $p(\theta)$ specifies our belief on the parameters before observing any data. This should capture any relevant information on the problem (if any) or being totally uninformative by letting "the data speak for itself". Such kind of priors (a. k. a. *uninformative priors*) are usually also improper as they don't integrate to one over their domain; examples are the *Haldane prior* (Zhu and A. Y. Lu, 2004) and the *Jeffreys prior* (Jeffreys, 1946). Note something very important: the Bayesian philosophy of reasoning about priors doesn't explain how to choose them (i.e., any prior can work with any model). In practice, some prior choices perform better than others, and we will see several examples of this in the rest of this thesis. The quantity $p(\mathbf{y} \,|\, f(\mathbf{X}; \theta))$ fixes our assumptions on the observed data (e.g. continuous and noisy, binary, count, etc.) and it measures how likely it is for the model f with parameters θ to have generated the observed values. This term—the *likelihood*—is the probability density function of the model given the parameters computed at the observed data. Examples of likelihoods are the *Gaussian likelihood* $\mathcal{N}(y \,|\, f(\mathbf{x}, \theta), \sigma^2)$ useful for regression tasks with i.i.d. noisy data, or the *Bernoulli likelihood* $\mathrm{Bern}\,(y \,|\, \lambda(f(\mathbf{x}, \theta)))$, for binary classification (with $\lambda : \mathbb{R} \to [0, 1]$).

2.1.1 Bayesian model selection

The denominator of Bayes' theorem is of particular interest. We refer to this as the *model evidence* or *marginal likelihood*, and its role can be better understood by explicitly conditioning the entire inference to a specific model hypothesis $\mathcal{M}$,

$$p(\theta \,|\, \mathbf{y}, \mathcal{M}) = \frac{p(\mathbf{y} \,|\, f(\mathbf{X}, \theta), \mathcal{M}) p(\theta \,|\, \mathcal{M})}{p(\mathbf{y} \,|\, \mathcal{M})}, \tag{2.2}$$

where $p(\mathbf{y} \,|\, \mathcal{M})$ is the normalization constant $\int p(\mathbf{y} \,|\, \theta, \mathcal{M}) p(\theta \,|\, \mathcal{M})\, d\theta$ required to have a proper definition of probability density function for the posterior. By comparing this quantity with the definition of the likelihood above, it becomes clear how $p(\mathbf{y} \,|\, \mathcal{M})$ measures how likely it is for the model $\mathcal{M}$ to have generated the observed data. It doesn't depend on the parameters θ of the model $\mathcal{M}$, but solely on the model hypothesis itself. Given this property, the marginal $p(\mathbf{y} \,|\, \mathcal{M})$ can be used for model comparison and model selection. This property of the marginal is also known as *Bayesian Occam' razor* effect (D. J. C. MacKay, 1992; Murray and Ghahramani, 2005): a principled way to model selection that doesn't incur into overfitting. A way to understand why the marginal likelihood follows the Occam's razor principle ("one should pick the simplest model that adequately explains the data") is by following the product rule of probabilities to rewrite this value,

$$p(\mathbf{y}) = \prod_{i=1}^{N} p(y_i \,|\, \mathbf{y}_{1:i-1}) = p(y_1) p(y_2 \,|\, y_1) p(y_3 \,|\, \mathbf{y}_{1:2}) \dots p(y_N \,|\, \mathbf{y}_{1:N-1}), \tag{2.3}$$

This highlights how the prediction for the next point $\mathbf{y}_i$ is conditioned to the previous ones $\mathbf{y}_{1:i-1}$, similarly to the leave-one-out cross-validation estimate of the likelihood (Murphy, 2012). Intuitively, if a model it too complex it will overfit with few examples and will perform poorly with the next ones.

We can further apply Bayes' rule to condition models on the data,

$$p(\mathcal{M} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathcal{M}) p(\mathcal{M})}{\int p(\mathbf{y}, \mathcal{M}) \, d\mathcal{M}} \tag{2.4}$$

where the right hand-side should be used in case of discrete model space. Solving such inference problem with complex systems (e.g. climate models, biological cell models, epidemiology, etc.) is challenging and it's a topic of an active area of research. The common and simple approach to this problem of model selection is the maximization of the marginal likelihood (also known as *Type-II maximum likelihood*). This can be further seen a maximum-a-posteriori (MAP) estimate of the posterior in Equation (2.4), when we place an uniform prior on the models $p(\mathcal{M}) \propto 1$. In practice, this is what happens when we optimize the kernel parameters for Gaussian processs (GPs) by maximization of the marginal likelihood.

The marginal likelihood has also an interpretation as generalization metric. Looking at the second axiom of probability theory, for a particular model hypothesis the marginal will sum to one over all possible datasets, e. g. $\sum_{\mathbf{y}'} p(\mathbf{y}' \mid \mathcal{M})$. Figure 2.1 gives an illustration of this property. For sake of clarity, suppose the dataset $\mathbf{y}$ to be a continuous random variable. Here, we plot the marginal likelihood as a function of the dataset for three different models with increasing complexity and we also indicate the observed data $\mathbf{y}'$. $\mathcal{M}_1$ is a simple model that performs well only on a very specific data that might not match the one that we observe. The low marginal corresponds to the low probability that indeed $\mathcal{M}_1$ has generated $\mathbf{y}'$. On the contrary, $\mathcal{M}_3$ is a very complex model that can predict many datasets; this means $\mathcal{M}_3$ assigns its probability thinly and widely across all possible data – making also this one unlikely to have generated $\mathbf{y}'$. Finally, $\mathcal{M}_2$ is just right as it performs well on $\mathbf{y}'$ with a reasonable confidence on its neighbor (Murphy, 2012; C. M. Bishop, 2006).

2.2 DEEP NEURAL NETWORKS

In the previous discussion, we kept the form of the model $f(\cdot)$ generic. In this section, we shall formalize the choice of neural networks, highlighting some important characteristics of this class of models and remarking the challenges for a Bayesian treatment.

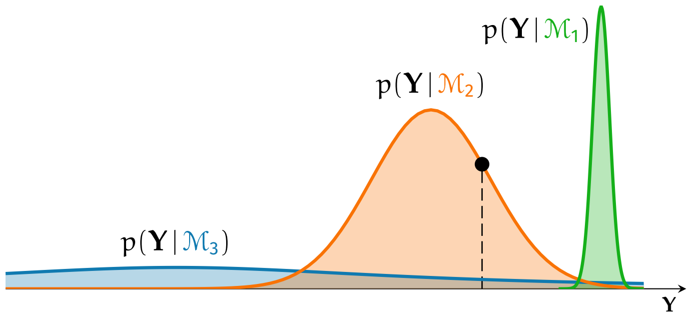


Figure 2.1: Graphical representation of the generalization interpretation of the marginal likelihood

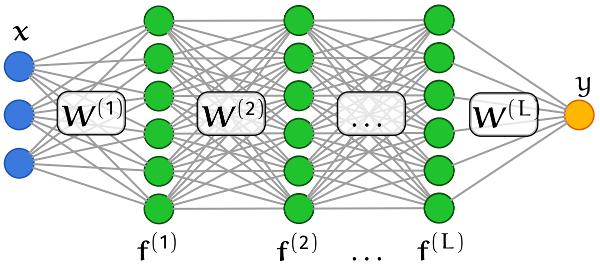


Figure 2.2: Example of a generic deep feedforward neural networks with L hidden layers. For visualization purposes the bias term is included in the notation $W^{(l)}$.

The basic idea behind deep neural networks is to use *composition of functions* to learn *hierarchical features representation* of data. The ability of learning and extracting useful features can be exploited for down-stream tasks, like regression or classification. *Deep feedforward neural networks*, also known as multilayer perceptrons (MLPs), are one of the most simple and popular architectures in deep learning. In feedforward neural networks (see an example in § 2.2), the data is processed by a chain of transformations, without the presence of any feedback loops. *Convolutional neural networks* are another example of feedforward models, where, differently from MLPs, weights are usually shared among different spatial patches of the input images. These architectures have proven to be very efficient in learning useful feature representations for highly structured data, like images, videos and text. Alternatively, architectures that implement loops are also known as *deep recurrent neural networks* and, despite their popularity for tasks like natural language processing (NLP), will not be discussed in this thesis.

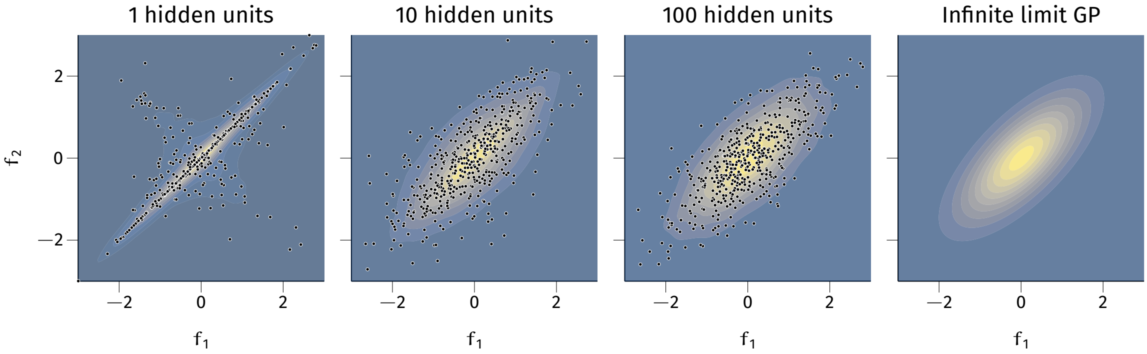


Figure 2.3: Analysis of the distribution of samples from the prior for a shallow neural network with tanh activation. Priors on the parameters are assumed to be Gaussian with zero mean and standard deviations of 5 for the weights and biases of the first layer, $D^{-0.5}$ and 0.1 for the weights and the bias on the second layer. The two random variables f_1 and f_2 are corresponding to inputs $x_1 = -0.2$ and $x_2 = 0.4$, respectively. The plots show 500 samples, while the density in the background is estimated with KDE and 10 000 samples.

2.2.1 Bayesian Neural Networks: Parameterization, Prior and Inference

Let's start by considering a deep neural network (DNN) consisting of L layers, where the output of the l-th layer $f_l(\mathbf{x})$ is a function of the previous layer outputs $f_{l-1}(\mathbf{x})$, as follows:

$$f_l(\mathbf{x}) = \frac{1}{\sqrt{D_{l-1}}}\Big(\mathbf{W}_l a(f_{l-1}(\mathbf{x}))\Big) + \mathbf{b}_l, \quad l \in \{1, \dots, L\}, \tag{2.5}$$

where $a(\cdot)$ is a nonlinearity, $\mathbf{b}_l \in \mathbb{R}^{D_l}$ is a vector containing the bias parameters for layer l, and $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l-1}}$ is the corresponding matrix of weights. In order to keep the notation uncluttered, we refer to the set of weights and bias parameters of a layer l as $\theta_l = \{W_l, b_l\}$, while the entirety of trainable network parameters is $\theta = \{\theta_l\}_{l=1}^{L}$. In order to simplify the presentation, we focus on fully-connected DNNs; note that the weight and bias parameters of convolutional neural networks (CNNs) can be treated in a similar way. To ensure the asymptotic variance of the output neither explodes nor vanishes, the output can be normalized with $\sqrt{D_{l-1}}$. For fully-connected layers, D_{l-1} is the dimension of the input, while for convolutional layers D_{l-1} is replaced with the filter size multiplied by the number of input channels.

The Bayesian treatment of neural networks dictates that a prior distribution $p(\theta)$ is placed over the parameters. Chapter 5 will be completely dedicated to the discussion on the role of prior distributions on Bayesian neural network (BNN). Nonetheless, it is important to comment on the asymptotic behavior of neural networks when we consider an infinite width limit. Neal (1996) observed that *"priors over network parameters can be defined in such a way that the corresponding priors over functions computed by the network reach reasonable limits as the number of hidden units goes to infinity"*. He then proves with the central limit theory that for shallow neural networks (i. e.one hidden layer) *"a Gaussian prior for hidden-to-output weights results in a Gaussian process prior for functions"*, meaning that for any set of input points $\mathbf{X}$, the random variables describing the functions computed at $\mathbf{X}$ have a joint Gaussian distribution (Rasmussen and Williams, 2005). In Figure 2.3 we show a pictorial representation of this phenomenon on a single hidden layer with tanh activation function and increasing width. This behavior has been observed in other deeper models (G. Matthews et al., 2018; Lee et al., 2018; M. E. E. Khan et al., 2019), for convolutional neural networks (Garriga-Alonso et al., 2019) and recurrent neural networks (G. Yang, 2019).

Recently this was also proved without the use of the central limit theory, by analyzing the effect of width in the prior marginal covariance (Pleiss and Cunningham, 2021).

The learning problem is formulated as a transformation of a prior belief into a posterior distribution by means of Bayes' theorem. Given a dataset with N input-target pairs $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} \overset{\text{def}}{=} \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, the posterior over θ is:

$$p(\theta \,|\, \mathbf{y}) = \frac{p(\mathbf{y} \,|\, \theta) p(\theta)}{p(\mathbf{y})}, \tag{2.6}$$

where the likelihood depends on the output of the neural network at input $\mathbf{X}$ and can generally be defined as $p(\mathbf{y} \,|\, f(\mathbf{X}; \theta))$. With the posterior inferred, we can make prediction on new test points $\mathbf{x}_\star$, which can be done my marginalizing out the parameters from the posterior,

For keeping the notation clean, we will avoid to explicit condition on the input $\mathbf{X}$.

$$p(y_\star \,|\, \mathbf{x}_\star, \mathbf{y}) = \int p(y_\star \,|\, \theta) p(\theta \,|\, \mathbf{y}) d\theta \tag{2.7}$$

While looking remarkably easy, Equations (2.6) and (2.7) hide several challenges: (i) the non-conjugacy of the likelihood and the prior makes such that we don't know the form of the posterior; (ii) the normalization constant at the dominator requires to compute the integral on the space of the parameters of the joint likelihood. As a consequence, the posterior on the parameters of a BNN is always analytically intractable and making predictions on new test points isnot possible, unless we resort to approximations. In Figure 2.4 we visualize the challenges of Bayesian

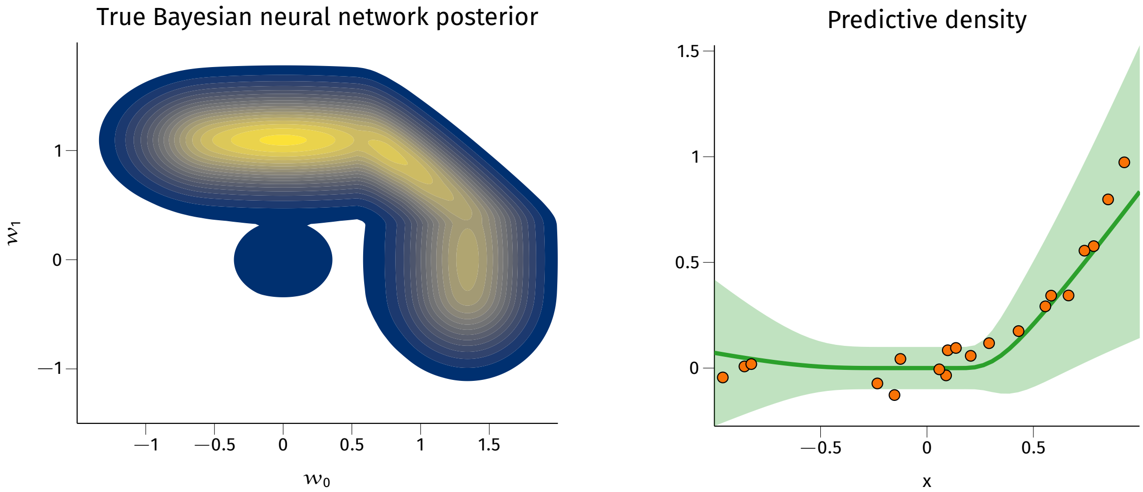


Figure 2.4: Pictorial representation of running Bayesian inference on a simple architecture $y = \mathrm{ReLu}(w_0 \cdot x - 0.5) + \mathrm{ReLu}(w_1 \cdot x - 0.3)$, where some parameters are fixed for visualization purposes. On the left, the posterior on the two parameters w_0 and w_1; on the right, the mean and the confidence intervals ($\pm 2\sigma$) of the predictive posterior $p(y_\star \,|\, \mathbf{x}_\star, \mathbf{y})$. In both cases we approximate all required integrals with numerical integration.

inference in neural networks with a simple case where we can run numerical integration to solve the various integrals involved. The visualization of the density highlights the non-trivial multimodal shape of the posterior; this gives some intuition on the huge challenges that approximate inference techniques have to deal with.

2.3 BAYESIAN INFERENCE AS OPTIMIZATION PROBLEM: VARIATIONAL INFERENCE

Variational inference (VI) is a classic tool to tackle intractable Bayesian inference (Jordan et al., 1999; D. M. Blei, Kucukelbir, et al., 2017). VI casts the inference problem into an optimization-based procedure to compute a tractable approximation of the true posterior. In our setting, we have a probabilistic model $p(\mathbf{y} \,|\, f(\mathbf{X}; \boldsymbol{\theta}))$ with parameters $\boldsymbol{\theta}$, a prior distributions on them $p(\boldsymbol{\theta})$ and a set of observations $\{\mathbf{X}, \mathbf{y}\}$. 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 $p(\boldsymbol{\theta} \,|\, \mathbf{y})$; and finally (iii) providing

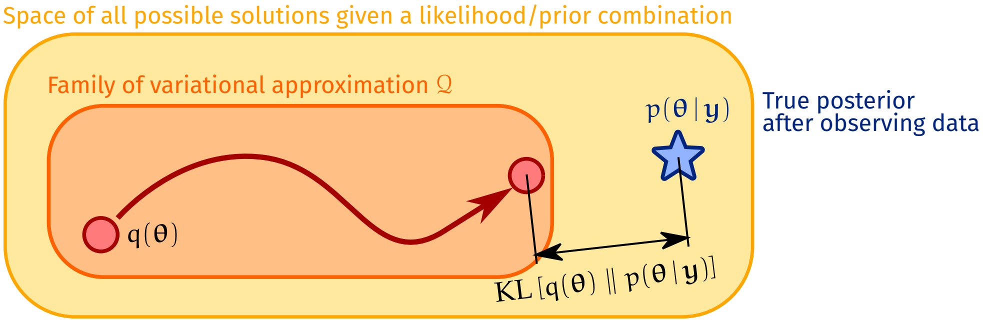

Space of all possible solutions given a likelihood/prior combination
Family of variational approximation $\mathcal{Q}$
$p(\theta \mid y)$
True posterior after observing data
$q(\theta)$
$\mathrm{KL}\left[q(\theta) \parallel p(\theta \mid y)\right]$

Figure 2.5: Pictorial illustration of the variational inference (VI) procedure, inspired by D. Blei et al. (2016).

a programmatic way to find the distribution $\tilde{q}(\theta)$ that minimizes such distance (Figure 2.5 provides a simple diagram of this procedure). In practice, $q(\theta)$ has some free parameters ν (also known as *variational parameters*), which are optimized such that the approximating distribution $q(\theta; \nu)$ is as closer as possible to the true posterior $p(\theta \mid y)$. The classical formulation of VI chooses the Kullback-Leibler (KL) divergence as a metric of similarity between these two distribution, but extensions to alternative divergences and distances are possible as well (Y. Li and Turner, 2016; Y. Li and Gal, 2017; Dieng et al., 2017; Wan et al., 2020). We can derive the variational objective starting from the definition of the KL,

The KL divergence between two distributions q *and* p *is* $\mathrm{KL}\left[q(z) \parallel p(z)\right] = \int q(z) \log \frac{q(z)}{p(z)} dz$.

The color red highlights the quantities that we cannot compute.

$$\begin{aligned}\mathrm{KL}\left[q(\theta;\nu) \parallel p(\theta \mid y)\right] &= \mathbb{E}_{q(\theta;\nu)}\left[\log q(\theta;\nu) - \log p(\theta \mid y)\right] = \\ &= \mathbb{E}_{q(\theta;\nu)}\left[\log q(\theta;\nu) - \log p(y \mid \theta) - \log p(\theta)\right] + \log p(y)\end{aligned} \tag{2.8}$$

Rearranging we have that

$$\log p(y) - \mathrm{KL}\left[q(\theta;\nu) \parallel p(\theta \mid y)\right] = \mathbb{E}_{q(\theta;\nu)}\left[\log q(\theta;\nu) - \log p(y \mid \theta) - \log p(\theta)\right] \tag{2.9}$$

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

$$\mathcal{L}_{\mathrm{ELBO}}(\nu) = \underbrace{\mathbb{E}_{q(\theta;\nu)} \log p(y \mid \theta)}_{\text{Model fitting term}} - \underbrace{\mathrm{KL}\left[q(\theta;\nu) \parallel p(\theta)\right]}_{\text{Regularization term}}. \tag{2.10}$$

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. We finally observe that minimizing the KL in Equation (2.8) is equivalent to find the variational parameters $\boldsymbol{\nu}$ such that the ELBO is maximized,

$$\tilde{\boldsymbol{\nu}} = \arg\max_{\boldsymbol{\nu}} \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\nu})} \log p(\mathbf{y} \mid \boldsymbol{\theta}) - \mathrm{KL}\left[q(\boldsymbol{\theta};\boldsymbol{\nu}) \parallel p(\boldsymbol{\theta})\right] \tag{2.11}$$

Before diving into the challenges of Equation (2.11), 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 (G. E. Hinton and Camp, 1993), 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,

$$q(\boldsymbol{\theta};\boldsymbol{\nu}) = \prod_{i=1}^{K} q(\theta_i;\boldsymbol{\nu}_i) \tag{2.12}$$

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,

$$q(\theta_i) = \mathcal{N}(\mu_i, \sigma_i^2) \tag{2.13}$$

Now, the collection of all means and variances $\{\mu_i, \sigma_i^2\}_{i=1}^K$ defines the set of variational parameters to optimize. In the next chapters we will discuss the limitations of simple approximations for Bayesian neural networks and how things can be improved.

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 (Metropolis and Ulam, 1949), by sampling N_{MC} times from $q_{\boldsymbol{\nu}}$,

$$\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\nu})} \log p(\mathbf{y} \mid \boldsymbol{\theta}) \approx \frac{1}{N_{\mathrm{MC}}} \sum_{j=1}^{N_{\mathrm{MC}}} \log p(\mathbf{y} \mid \tilde{\boldsymbol{\theta}}_j), \quad \text{with} \quad \tilde{\boldsymbol{\theta}}_j \sim q(\boldsymbol{\theta};\boldsymbol{\nu}) \tag{2.14}$$

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. Despite its simplicity, the estimation in Equation (2.14) is unbiased and its variance asymptotically shrinks with $\frac{1}{N_{\mathrm{MC}}}$, independently of the dimensionality of $\boldsymbol{\theta}$.

GOODNESS OF THE APPROXIMATION. Equation (2.9) allows us to understand the relationship between the marginal likelihood and the (negative) evidence lower bound. In the ideal case of exact solution for the posterior, equivalent to having $\mathrm{KL}\left[q(\theta;\nu) \,\|\, p(\theta \,|\, y)\right] = 0$, the minimization of the ELBO is equivalent to the maximization of the marginal likelihood, with all the properties analyzed in § 2.1.1. In the general case, the ELBO is just a lower bound of the marginal and the gap between these two quantities depends on the level of approximation of q. To narrow this gap and therefore obtain a tighter lower bound, we need to increase the expressiveness of the variational posterior.

2.3.1 Optimization of the ELBO

We now have a tractable objective that needs to be optimized with respect to the variational parameters ν. 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 differentiation falls in the category of taking gradients of a quadrature, i. e.$\nabla_\nu \mathbb{E}_{q(\theta;\nu)} f(\theta)$. An unbiased estimate of such gradient can be derived as follows,

$$\nabla_\nu \mathbb{E}_q f(\theta) = \mathbb{E}_q \left[f(\theta) \nabla_\nu \log q(\theta)\right] \tag{2.15}$$

which can be estimated again by sampling from q. The full gradient of the ELBO can be computed starting from Equation (6.2) as follows,

$$\nabla_\nu \mathcal{L}_{\text{ELBO}} = \mathbb{E}_q \left[p_\theta(y \,|\, \theta) \nabla_\nu \log q_\nu(\theta)\right] - \nabla_\nu \mathrm{KL}\left[q_\nu(\theta) \,\|\, p(\theta)\right] . \tag{2.16}$$

This estimator has a very general formulation that make it applicable to a wide range of situations. In the literature, this is also refer as *score function estimator* or *black-box variational inference* (D. M. Blei, Jordan, et al., 2012), due to the fact that it doesn't require to take the derivative of the model likelihood $p(y \,|\, \theta)$. As such, the *score function estimator* can be used also for non-differentiable likelihoods. On the other hand, this estimator shows huge variance and, unless remedies like control variates (Ross, 2006) are taken, this estimator is impractical. Given that most of BNNs architectures (if not all) are easily differentiable, there are not practical uses for this formulation in Bayesian deep learning.

Alternatively, this problem can be solved using the so-called *reparameterization trick* (Salimans and Knowles, 2013; Kingma and Welling, 2014). The reparameterization trick aims at constructing θ as an invertible function $\mathcal{T}$ of the variational parameters $\boldsymbol{\nu}$ and of another random variable ε, so that $\theta = \mathcal{T}(\varepsilon; \boldsymbol{\nu})$. ε is chosen such that its marginal $p(\varepsilon)$ 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,

Generally, a $\mathcal{T}$ that suits this constraint might not exists; Ruiz et al. (2016) 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 (Maddison et al., 2017).

$$\theta \sim \mathcal{N}(\mu, \sigma^2) \iff \theta = \mu + \sigma\varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, 1)\,. \tag{2.17}$$

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

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p(\varepsilon)} \left[\nabla_{\theta} \log p(\mathbf{y} \,|\, \theta) \,|_{\theta = \mathcal{T}(\varepsilon; \boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \mathcal{T}(\varepsilon; \boldsymbol{\nu}) \right] - \nabla_{\boldsymbol{\nu}} \mathrm{KL} \left[q(\theta; \boldsymbol{\nu}) \,\|\, p(\theta) \right]\,. \tag{2.18}$$

The gradient $\nabla_{\theta} \log p(\mathbf{y} \,|\, \theta)$ depends on the model and it can be derived with automatic differentation tools (Abadi et al., 2015; Paszke et al., 2019), while $\nabla_{\boldsymbol{\nu}} \mathcal{T}(\varepsilon; \boldsymbol{\nu})$ 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,

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p(\varepsilon)} \left[\nabla_{\theta} \log p(\mathbf{y} \,|\, \theta) + \log q(\theta; \boldsymbol{\nu}) - \log p(\theta) \right]_{\theta = \mathcal{T}(\varepsilon; \boldsymbol{\nu})} \nabla_{\boldsymbol{\nu}} \mathcal{T}(\varepsilon; \boldsymbol{\nu}) \tag{2.19}$$

Roeder et al. (2017) argue that when we believe that $q(\theta; \boldsymbol{\nu}) \approx p(\mathbf{y} \,|\, \theta)$, Equation (2.19) should be prefered over Equation (2.18) 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 Equation (2.18) can be computationally challenging, due to the evaluation of the likelihood and its gradients N_{MC} times. Assuming factorization of the likelihood,

$$p(\mathbf{y} \,|\, \theta) = p(\mathbf{y} \,|\, f(\mathbf{X}; \theta)) = \prod_{i=1}^{N} p(y_i \,|\, f(\mathbf{x}_i; \theta)) \tag{2.20}$$

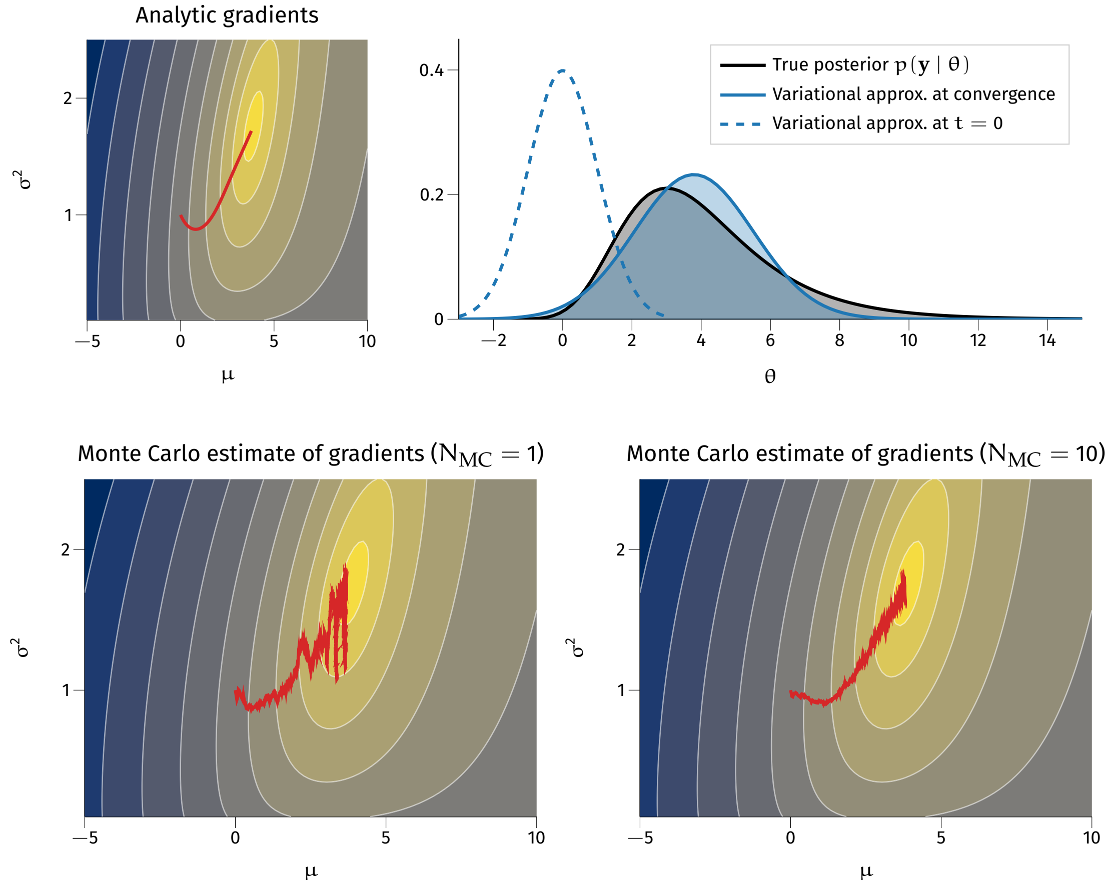


Figure 2.6: Animation of a simple variational inference procedure, using analytical gradients and Monte Carlo estimation.

this quantity can be approximated using mini-batching (Graves, 2011; Hoffman et al., 2013). Recalling $\mathbf{y}$ as the set of labels of our dataset with N examples, by taking $\mathcal{B} \subset \mathbf{y}$ as a random subset of $\mathbf{y}$, the likelihood term can be estimated in an unbiased way as

$$\log p_{\theta}(\mathbf{y} \mid \theta) \approx \frac{N}{M} \sum_{y_i \sim \mathcal{B}} \log p(y_i \mid \theta) \,. \tag{2.21}$$

where M is the number of points in the minibatch. At the cost of increase "randomness", we can use Equation (2.18) to compute the gradients of the ELBO with the minibatch formulation in Equation (2.21). Stochastic optimization, e.g. any version of stochastic gradient descent (SGD), will converge to a local optimum provided with a decreasing learning rate and sufficient gradient updates (Robbins and Monro, 1951).

This procedure can be further improved by analyzing the magnitude of the noise for the stochastic gradients. Focusing on the likelihood in Equation (2.21), let's define ℓ_i the likelihood contribution $\log p(y_i \,|\, \theta)$ from the datapoint y_i. From here, we can derive the variance of this term by considering the minibatch to be drawn randomly with replacement from $\mathbf{y}$ (Kingma and Ba, 2015),

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\mathbb{E}_{q(\theta;\mathbf{v})} \log p(\mathbf{y} \,|\, \theta)\right] &= \frac{N^2}{M^2}\left(\sum_{i=0}^{M} \mathbb{V}\mathrm{ar}\left[\ell_i\right] + 2\sum_{i=0}^{M}\sum_{j=i+1}^{M} \mathbb{C}\mathrm{ov}\left[\ell_i, \ell_j\right]\right) \\ &\approx N^2\left(\frac{1}{M}\mathbb{V}\mathrm{ar}\left[\ell_i\right] + \frac{M-1}{M}\mathbb{C}\mathrm{ov}\left[\ell_i, \ell_j\right]\right)\end{aligned} \tag{2.22}$$

where the $\mathbb{V}\mathrm{ar}\left[\cdot\right]$ and the $\mathbb{C}\mathrm{ov}\left[\cdot\right]$ are taken with respect to the empirical distribution defined by the dataset and the distribution of θ. As we can see, the variance term decreases linearly with M but the total contribution does not scale with the number of points in the minibatch. This is due to the correlation term which is asymptotically independent on the minibatch size. In practice, this term can dominate the whole variance of the likelihood even for big minibatch. This can be alleviated by imposing independence between cross-terms in the likelihood, i.e. $\mathbb{C}\mathrm{ov}\left[\ell_i, \ell_j\right] = 0$. This is achieved by imposing independence between input points and the samples from the distribution $q_{\mathbf{v}}(\theta)$; in practice, one would need to resample θ exactly N_{MC} times for each point y_i. This is generally computationally challenging but for BNNs it becomes as easy as moving the randomness from the weights to the pre-activations. Take as an example the setup of Figure 2.7 where for conveniance we defined $\mathbf{A}$ as the pre-activation map for the first hidden layer of a mini-batch of data $\mathbf{X}$. Given our choice of factorized Gaussian on the weights $q(W_{ij}) = \mathcal{N}(\mu_{ij}, \sigma^2_{ij})$, we can write down the expression for the distribution over $\mathbf{A}$, which has the following form,

$$q(\mathbf{A};\mathbf{v}) = \prod_{m=1}^{M}\prod_{j=1}^{D_1} \mathcal{N}\left(\sum_{k=1}^{D_0} X_{nk}\mu_{kj}, \sum_{k=1}^{D_0} X^2_{nk}\sigma^2_{kj}\right) \tag{2.23}$$

This new estimator is also known as the *local reparameterization trick* (Kingma and Ba, 2015), if otherwise stated it will be the default choice for variational inference on BNNs.

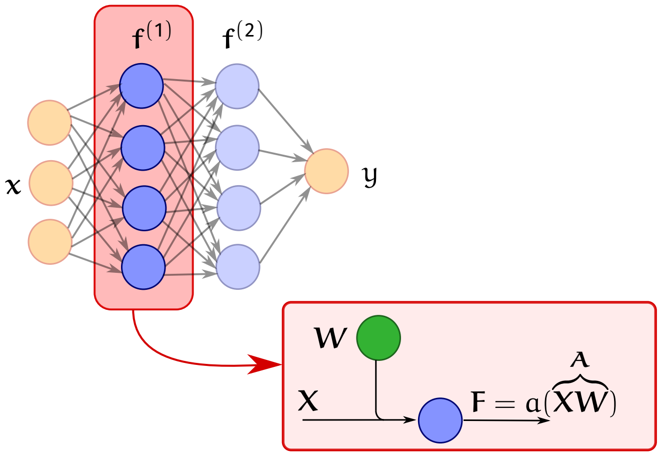


Figure 2.7: Graphical representation of the reparameterization trick.

2.4 SAMPLING WITH SCALABLE MARKOV CHAIN MONTE CARLO

Hamiltonian Monte Carlo (HMC) methods provide an elegant and efficient way to sample from intractable distributions by defining proposals with probability one of being accepted in the classic Metropolis-Hastings framework. These proposals are generated by simulating an Hamiltonian system where the potential energy U is our intractable (log-)posterior density $-\log p(\theta \mid y)$ and the kinetic energy is parameterized with some auxilary momentum variables ρ. To propose samples, HMC simulates the following dynamics, derived from the theory of Hamiltonian systems (Hamilton-Jacobi equations):

$$d\theta = \mathbf{M}^{-1}\rho dt \tag{2.24}$$

$$d\rho = \nabla U(\theta) dt \tag{2.25}$$

While the simulation of this continuous system is often intractable, the classic recipe calls for discretization using e.g. the *leapfrog* scheme and correction step using the classic MH acceptance/rejection framework. In this case, the MH works on the Hamiltonian function, which is defined as

$$H(\theta, \rho) = U(\theta) + \frac{1}{2}\rho^{\top}\mathbf{M}^{-1}\rho \tag{2.26}$$

Algorithm 1 sketches the HMC sampler, including the steps required for the discretization of the dynamics.

Algorithm 1: Hamiltonian Monte Carlo

```
Input: Starting position $\theta^{(1)}$ and step size $\varepsilon$
for $t = 1, 2 \cdots$ do
    Resample momentum:
    $\rho^{(t)} \sim \mathcal{N}(0, \mathbf{M})$
    $(\theta_0, \rho_0) = (\theta^{(t)}, \rho^{(t)})$
    Leapfrog discretization of Hamiltonian dynamics in Equation (2.24):
    $\rho_0 \leftarrow \rho_0 - \frac{\varepsilon}{2} \nabla U(\theta_0)$
    for $i = 1$ to $m$ do
        $\theta_i \leftarrow \theta_{i-1} + \varepsilon \mathbf{M}^{-1} \rho_{i-1}$
        $\rho_i \leftarrow \rho_{i-1} - \varepsilon \nabla U(\theta_i)$
    $\rho_m \leftarrow \rho_m - \frac{\varepsilon}{2} \nabla U(\theta_m)$
    $(\hat{\theta}, \hat{\rho}) = (\theta_m, \rho_m)$
    Metropolis-Hastings correction:
    $u \sim \mathrm{Uniform}(0, 1)$
    $\rho = e^{H(\hat{\theta}, \hat{\rho}) - H(\theta^{(t)}, \rho^{(t)})}$
    if $u < \min(1, \rho)$, then $\theta^{(t+1)} = \hat{\theta}$
```

This way of proposing samples is very efficient but computationally challenging, as it requires to compute the gradients of the unnormalized log-posterior. While the prior component is usually not a problem, the evaluation of the gradient of the likelihood is, especially for large datasets. If we suppose the likelihood to factorize on the observations, we can approximate this quantity using mini-batches:

$$\nabla \widetilde{U}(\theta) = -\frac{N}{M} \sum_{i=1}^{M} \nabla \log p(y_i \mid \theta) - \nabla \log p(\theta) \tag{2.27}$$

Using mini-batches, we can write the gradients as unbiased estimates of the original $U(\theta)$ as follows,

$$\nabla \widetilde{U}(\theta) = \nabla U(\theta) + \mathcal{N}(0, \mathbf{V}(\theta)) \tag{2.28}$$

where $\mathbf{V}(\theta)$ is the covariance of the stochastic gradient noise, which depends on many factors including the parameters themselves.

A first solution to implement Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) is to replace the computation of the gradients in Algorithm 1 with Equation (2.27). Unfortunately, this is not enough: as T. Chen et al. (2014) proves, the simulation of the resulting dynamics would make the stationary distribution no longer invariant (i.e., sampling is performed on a distribution which is not the original $p(\theta \mid y)$).

To solve this problem, T. Chen et al. (2014) proposes to modify the dynamics by introducing an additional *friction* term $\mathbf{C}$. With this addition, the discretized system of equation becomes,

$$\Delta\theta = \varepsilon \mathbf{M}^{-1}\rho \tag{2.29}$$

$$\Delta\rho = \varepsilon \nabla \widetilde{U}(\theta) - \varepsilon \mathbf{C}\mathbf{M}^{-1}\rho + \mathcal{N}(0, 2(\mathbf{C} - \widehat{\mathbf{B}})\varepsilon) \tag{2.30}$$

where $\widehat{\mathbf{B}}$ is an estimation of the stochastic gradient noise covariance.

From a practical point of view, this system of dynamics is still challenging to work with. Namely, one should choose the friction term $\mathbf{C}$, perform an estimation of the noise gradient $\widehat{\mathbf{B}}$ and choose the mass matrix $\mathbf{M}$ and the step-size ε. While the friction term and the step-size are high model and dataset dependent, the other two quantities can be estimated during the burn-in phase (Springenberg et al., 2016). For the mass matrix, we can leverage the connection between SGHMC and SGD. Results in stochastic optimization literature (Duchi et al., 2011; Tieleman and G. Hinton, 2012) discuss how the robustness of SGD can be improved by normalizing the gradient by its magnitude. For SGHMC this is equivalent to choosing,

$$\mathbf{M}^{-1} = \operatorname{diag}\left(\widehat{\mathbf{V}}_\theta^{-\frac{1}{2}}\right) \tag{2.31}$$

where $\widehat{\mathbf{V}}_\theta$ is the estimation of the element-wise variance of the gradient and it can be estimated by exponential moving average as follow,

$$\Delta\widehat{\mathbf{V}}_\theta = -\tau^{-1}\widehat{\mathbf{V}}_\theta + \tau^{-1}\nabla(U(\theta))^2 \tag{2.32}$$

The averaging window is specified by the choice of τ, which–again–can be automatically determined by using a procedure similar to adaptive learning rate for SGD (Tieleman and G. Hinton, 2012),

$$\Delta\tau = -g_\theta^2 \widehat{\mathbf{V}}_\theta^{-1}\tau + 1 \tag{2.33}$$

$$\Delta g_\theta = -\tau^{-1}g_\theta + \tau^{-1}\nabla\widetilde{U}(\theta) \tag{2.34}$$

where $\mathfrak{g}_\theta$ is a smoothed estimation of the gradients $\boldsymbol{\nabla}U(\theta)$. Finally, we can also use $\widehat{\mathbf{V}}_\theta$ for $\widehat{\mathbf{B}}$, specifically $\widehat{\mathbf{B}} = \frac{1}{2}\widehat{\mathbf{V}}_\theta \varepsilon$. By substituting these new quantities in the original dynamics we obtain

$$\Delta\theta = \boldsymbol{\nu} \tag{2.35}$$

$$\Delta\boldsymbol{\nu} = -\varepsilon^2 \widehat{\mathbf{V}}_\theta^{-\frac{1}{2}} \boldsymbol{\nabla}\widetilde{U}(\theta) - \varepsilon \widehat{\mathbf{V}}_\theta^{-\frac{1}{2}} \mathbf{C}\boldsymbol{\nu} + \mathcal{N}\left(0, 2\varepsilon^3 \widehat{\mathbf{V}}_\theta^{-\frac{1}{2}} \mathbf{C} \widehat{\mathbf{V}}_\theta^{-\frac{1}{2}} - \varepsilon^4 \mathbf{I}\right) \tag{2.36}$$

where $\boldsymbol{\nu}$ is defined as $\boldsymbol{\nu} = \varepsilon \mathbf{M}^{-1} \boldsymbol{\rho} = \varepsilon \widehat{\mathbf{V}}_\theta^{-\frac{1}{2}} \boldsymbol{\rho}$.

3

INITIALIZATIONS OF VARIATIONAL INFERENCE FOR BAYESIAN NEURAL NETWORKS

Stochastic variational inference is an established way to carry out approximate Bayesian inference for deep models flexibly and at scale. While there have been effective proposals for good initializations for loss minimization in deep learning, far less attention has been devoted to the issue of initialization of stochastic variational inference. In this chapter we analyze this problem for Bayesian deep neural network and we propose a novel layer-wise initialization strategy based on Bayesian linear models. The proposed method is extensively validated on regression and classification tasks, including Bayesian deep neural networks and Bayesian convolutional neural networks, showing faster and better convergence compared to alternatives inspired by the literature on initializations for loss minimization.

3.1 OVERVIEW

Deep neural networks (DNNS) and convolutional neural networks (CNNS) have become the preferred choice to tackle various learning tasks, due to their ability to model complex problems and the mature development of regularization techniques to control overfitting (LeCun, Bengio, et al., 2015; Srivastava et al., 2014). There has been a recent surge of interest in the issues associated with their over-confidence in predictions, and proposals to mitigate these (Guo et al., 2017; Kendall and Gal, 2017; Lakshminarayanan et al., 2017). Bayesian techniques offer a natural framework to

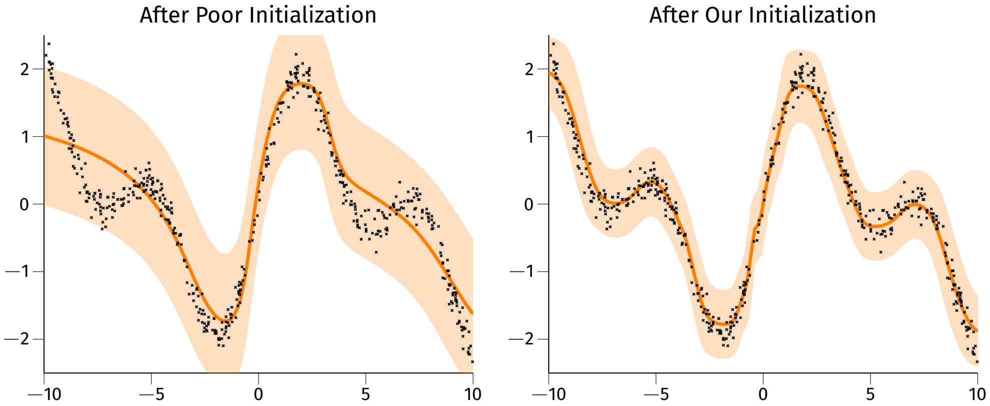


Figure 3.1: Due to poor initialization (**left**) VI fails to converge even after 1500+ epochs (RMSE = 0.434, NLL = 257.38) while with our iterative Bayesian linear modeling (IBLM) (**right**) VI easily recovers the function after fewer epochs (RMSE = 0.219, NLL = -88.32). The architecture has three hidden layers with 500 neurons each, and uses the TANH activation function.

deal with such issues, but they are characterized by computational intractability (C. M. Bishop, 2006; Ghahramani, 2015). A popular way to recover tractability is to use variational inference (Jordan et al., 1999). In variational inference (VI), an approximate posterior distribution is introduced and its parameters are adapted by optimizing a variational objective, which is a lower bound to the marginal likelihood. Stochastic variational inference offers a practical way to carry out stochastic optimization of the variational objective. Here stochasticity is introduced with a doubly stochastic approximation of the expectation term, which is unbiasedly approximated using Monte Carlo and by selecting a subset of the training points (Graves, 2011; Kingma and Welling, 2014). While VI is an attractive and practical way to perform approximate inference for DNNs, there are limitations. For example, the form of the approximating distribution can be too simple to accurately approximate complex posterior distributions (Ha et al., 2016; Ranganath et al., 2015; D. Rezende and Mohamed, 2015). Furthermore, VI increases the number of optimization parameters compared to optimizing model parameters through, e.g., loss minimization; for example, a fully factorized Gaussian posterior over model parameters doubles the number of parameters in the optimization compared to loss minimization. This has motivated research on other ways to perform approximate Bayesian inference for DNNs by establishing connections between variational inference and dropout (Gal and Ghahramani, 2016b; Gal and Ghahramani, 2016a; Gal, Hron, et al., 2017).

A theoretical understanding of the optimization landscape of DNNs and CNNs is still in its early stages of development (Dziugaite and Roy, 2017; Garipov et al., 2018), and most works have focused on the practical aspects characterizing the optimization of their parameters (Duchi et al., 2011; Kingma and Ba, 2015; Srivastava et al., 2014). If this lack of theory is apparent for optimization of model parameters, this is even more so for the understanding of the optimization landscape of the objective in variational inference, where variational parameters enter in a nontrivial way in the objective (Graves, 2011; D. J. Rezende et al., 2014). Initialization plays a huge role in the convergence of VI; the illustrative example in Figure 3.1 shows how a poor initialization can prevent VI to converge to good solutions in short amount of time. The problem is even more severe for complex architectures, such as the ones that we discuss in the experiments; for example, VI systematically converges to trivial solutions (posterior equal to the prior) when applied to CNNs. In this chapter, we focus on this issue affecting VI for DNNs and CNNs. While there is an established literature on ways to initialize model parameters of DNNs when minimizing its loss (Glorot and Bengio, 2010; Saxe et al., 2013; Mishkin and Matas, 2016), to the best of our knowledge, there is no study that systematically tackles this issue for VI for Bayesian DNNs and CNNs. Inspired by the literature on residual networks (K. He et al., 2016) and greedy initialization of DNNs (Bengio, Lamblin, et al., 2006; Mishkin and Matas, 2016), we propose a novel initialization strategy for VI grounded on Bayesian linear modeling, which we call IBLM. Iterating from the first layer, IBLM initializes the posterior at layer (l) by learning Bayesian linear models which regress from the input, propagated up to layer (l), to the labels.

We show how IBLM can be applied in a scalable way and without considerable overhead to regression and classification problems, and how it can be applied to initialize VI not only for DNNs but also for CNNs. Through a series of experiments, we demonstrate that IBLM leads to faster convergence compared to other initializations inspired by prior work on loss minimization for DNNs. Furthermore, we show that IBLM makes it possible for VI with a Gaussian approximation applied to CNNs to compete with Monte Carlo dropout (MCD) (Gal and Ghahramani, 2016a) and noisy natural gradients (NOISY-KFAC; G. Zhang et al. (2018)), which are state-of-art methods to perform approximate inference for CNNs.

In summary, in this chapter we make the following contributions:

- we propose a novel way to initialize VI for DNNs based on Bayesian linear models;

- we show how this can be done for regression and classification;
- we show how to apply our strategy to CNNs;
- we empirically demonstrate that our proposal allows us to achieve performance superior to other initializations of VI inspired by the literature on loss minimization;
- for the first time, we are able to run Gaussian VI on large-scale CNNs, obtaining remarkable performances.

3.1.1 A review of the role of initialization in deep learning

The problem of initialization of weights and biases in DNNs for gradient-based loss minimization has been extensively tackled in the literature since early breakthroughs in the field (Rumelhart et al., 1986; Baldi and Hornik, 1989). Rumelhart et al. (1986) study the limitations of backpropagation-based gradient descend to find good local minima in the parameter space, showing that this issue arises even in small DNNs; the authors address this issue by initalizing the optimization using random weights. Few years later, Baldi and Hornik (1989) report that for layered linear feed-forward neural networks trained using a quadratic loss function, there exists a unique global minimum corresponding to a orthogonal projection of the ordinary least square solution into a subspace spanned by eigenvalues of the covariance matrix of training samples. Nevertheless, the restrictions required to derive this closed-form solution (linear layers without bias on a single hidden layer networks with equal number of input and output features) severely limit the applicability of these results to a wider and more interesting class of DNNs. To overcome these limitations, LeCun, Bottou, et al. (1998) discusses practical tricks to achieve an efficient loss minimization through back-propagation; choosing separate learning rate for each weight and a Gaussian random initialization are some of the proposed ways to obtain an effective optimization.

More recently, Bengio, Lamblin, et al. (2006) propose a greedy layer-wise unsupervised pre-training, which proved to help optimization and generalization. A justification can be found in Erhan et al. (2010), where the authors show that pre-training can act as regularization; by initializing the parameters in a region corresponding to a better basin of attraction for the optimization procedure, the model can reach a better local minimum and increase its generalization capabilities. Glorot and Bengio (2010) propose a simple way to estimate the variance for random initialization of weights, which makes it possible to avoid saturation both in forward and back-propagation steps. Another possible strategy can be found in the work by Saxe et al. (2013), that investigates the dynamics of gradient descend optimization, and proposes a random orthogonal initialization of the weights based on the singular value decomposition of a Gaussian random matrix. This algorithm takes a random weight matrix filled with Gaussian noise, decomposes it to orthonormal basis using a singular value decomposition and replaces the weights with one of the components. Building on this work, Mishkin and Matas (2016) propose a data-driven weight initialization by scaling the orthonormal matrix of weights to make the variance of the output as close to one as possible.

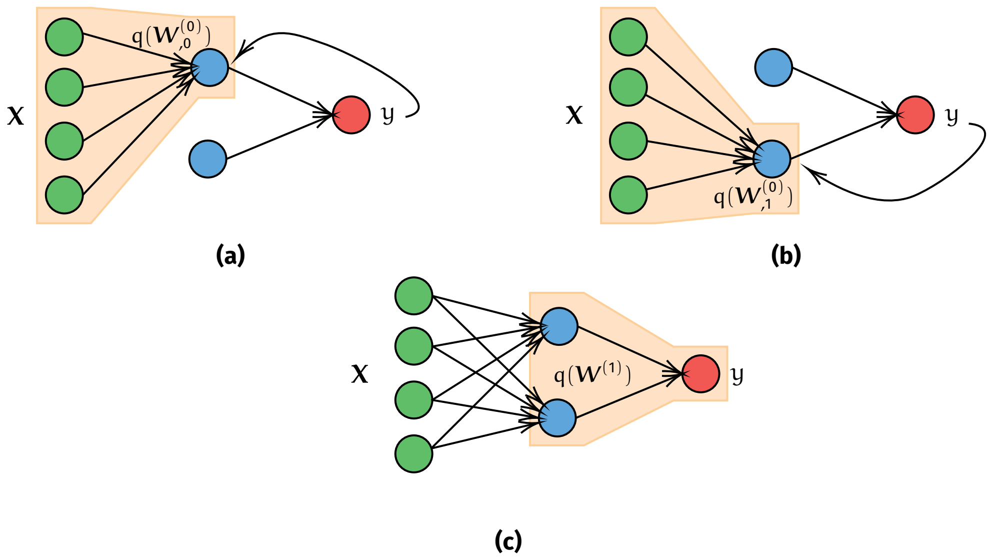
Variational inference addresses the problem of intractable Bayesian inference by reinterpreting inference as an optimization problem. The reparameterization trick (Kingma and Welling, 2014) allows for the stochastic optimization of the variational lower bound through automatic differentiation (Duchi et al., 2011; Zeiler, 2012; Sutskever et al., 2013; Kingma and Ba, 2015). Despite the tight connection between loss minimization and evidence lower bound (ELBO) maximization (Graves, 2011), to the best of our knowledge there is no study that either empirically or theoretically addresses the problem of initialization of parameters for VI; we could only find a mention of this in Krishnan et al. (2018) for variational autoencoders. We aim to fill this gap by proposing a novel way to initialize parameters in VI for probabilistic deep models.

3.2 INITIALIZATION OF VARIATIONAL PARAMETERS: A PROPOSED METHOD

In this section, we introduce our proposed Iterative Bayesian Linear Model (IBLM) initialization for VI. We first introduce IBLM for regression with DNNs, and we then show how this can be extended to classification and to CNNs.

3.2.1 Initialization of DNNs for Regression

In order to initialize the weights of DNNs, we proceed iteratively as follows. Before applying the nonlinearity through the activation function, each layer in a Bayesian DNN can be seen as multivariate Bayesian linear regression model. We use this observation as an inspiration to initialize the variational parameters



(a) (b) (c)

Figure 3.2: Visual representation of the proposed method for initialization. In (a) and (b), we learn two Bayesian linear models, whose outputs are used in (c) to infer the following layer.

as follows. Starting from the first layer, we can set the parameters of $q(W^{(0)})$ by running Bayesian linear regression with inputs X and labels y. Let's considering for a moment a particular column j of the weights matrix, which we denote with $w = W_{\cdot j}^{(0)}$. By choosing a Gaussian prior and a Gaussian likelihood as follows

$$p(w) = \mathcal{N}(w \mid 0, \Lambda) \tag{3.1}$$

$$p(y \mid w, X) = \mathcal{N}(y \mid Xw, L) \tag{3.2}$$

the posterior $p_{lm}(w \mid y, X)$ as simple as apply the rule of Gaussian random variables (C. M. Bishop, 2006),

$$p_{lm}(w \mid y, X) = \mathcal{N}(w \mid m, S) = \mathcal{N}(w \mid SX^\top L^{-1} y, S)\,, \tag{3.3}$$

where $S^{-1} = \Lambda^{-1} + X^\top L^{-1} X$. This process is eventually repeating the process for multiple hidden units (the posterior factorizes on the columns of $W^{(0)}$). After this, we initialize the approximate posterior over the weights at the second layer $q(W^{(1)})$ by running Bayesian linear regression with inputs $X = a(X\widetilde{W}^{(0)})$ and labels y, where $\widetilde{W}^{(0)}$ is a sample from $q(W^{(0)})$. We then proceed iteratively in the same way up to the last layer. Figure 3.2 gives an illustration of the proposed method for a simple architecture.

Remember that $a(\cdot)$ denotes for us the element-wise application of the activation function to the argument

The intuition behind IBLM is as follows. If one layer is enough to capture the complexity of a regression task, we expect to be able to learn an effective mapping right after the initialization of the first layer. In this case, we also expect that the mapping at the next layers implements simple transformations, close to the identity. Learning a set of weights with these characteristics starting from a random initialization is far from trivial, which also motivated the work on residual networks (K. He et al., 2016). Our IBLM initialization takes this observation as an intuition to initialize VI for general deep models.

From a complexity point of view, denoting by D_l the number of output neurons at layer (l), this is equivalent to D_l univariate Bayesian linear models. Instead of using the entire training set to learn the linear models, each one of these is inferred based on a random mini-batch of data, whose inputs are propagated through the previous layers. The complexity of IBLM is linear in the batch size and cubic in the number of neurons to be initialized. Later on, we will provide an evaluation of the effect of batch size and a timing profiling of IBLM.

3.2.2 From the Bayesian linear model posterior to the variational approximation

The proposed IBLM initialization of variational parameters can be used with any choice for the form of the approximate posterior. The exact posterior of Bayesian linear regression is not factorized, so one needs to match this with the form of the chosen approximate posterior. For simplicity of notation, let $\boldsymbol{w}$ be one of the parameters vector of interest. We can formulate this problem by minimizing the Kullback-Leibler (KL) divergence from the variational approximation $q(\boldsymbol{w}; \boldsymbol{\nu})$ to the posterior obtained from the linear model $p_{lm}(\boldsymbol{w} \,|\, \mathbf{X}, \boldsymbol{y})$. In the case of a fully factorized approximate posterior over the weights $q(\boldsymbol{w}; \boldsymbol{\nu}) = \mathcal{N}(\boldsymbol{w} \,|\, \boldsymbol{\mu}, \mathbf{I}\boldsymbol{\sigma}^2)$, this minimization can be done analytically.

Recall that the expression for the KL divergence between two multivariate Gaussians $p_0 = \mathcal{N}(\boldsymbol{w} \,|\, \boldsymbol{\mu}_0, \Sigma_0)$ and $p_1 = \mathcal{N}(\boldsymbol{w} \,|\, \boldsymbol{\mu}_1, \Sigma_1)$ is

$$\mathrm{KL}\left[p_0 \,\|\, p_1\right] = \frac{1}{2}\operatorname{Tr}(\Sigma_1^{-1}\Sigma_0) + \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\top}\Sigma_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - \frac{D}{2} + \frac{1}{2}\log\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right) \quad (3.4)$$

The KL divergence is not symmetric, so the order in which we take this matters. If we consider $\mathrm{KL}\left[q(\boldsymbol{w}) \parallel p_{lm}(\boldsymbol{w}|\mathbf{X}, \mathbf{y})\right]$, the expression becomes:

$$\mathrm{KL}\left[q(\boldsymbol{w}) \parallel p(\boldsymbol{w}|\mathbf{X}, \mathbf{y})\right] = \frac{1}{2}\mathrm{Tr}\big(\mathbf{S}^{-1}\mathrm{diag}(\boldsymbol{\sigma}^2)\big) + \frac{1}{2}(\mathbf{m}-\boldsymbol{\mu})^\top \mathbf{S}^{-1}(\mathbf{m}-\boldsymbol{\mu}) - \frac{D}{2} + \frac{1}{2}\log\left(\frac{\det \mathbf{S}}{\prod_i \sigma_i^2}\right) \tag{3.5}$$

It is a simple matter to show that the optimal mean $\boldsymbol{\mu}$ is $\mathbf{m}$ as $\boldsymbol{\mu}$ appears only in the quadratic form which is clearly minimized when $\boldsymbol{\mu} = \mathbf{m}$. For the variances $\boldsymbol{\sigma}^2$, we need to take the derivative of the KL divergence and set it to zero:

$$\frac{\partial \mathrm{KL}\left[q(\boldsymbol{w}) \parallel p(\boldsymbol{w}|\mathbf{X}, \mathbf{y})\right]}{\partial \sigma_i^2} = \frac{1}{2}\frac{\partial\, \mathrm{Tr}\big(\mathbf{S}^{-1}\mathrm{diag}(\boldsymbol{\sigma}^2)\big)}{\partial \sigma_i^2} - \frac{1}{2}\frac{\partial \sum_i \log \sigma_i^2}{\partial \sigma_i^2} = 0 \tag{3.6}$$

Rewriting the trace term as the sum of the Hadamard product of the matrices in the product $\sum_{ij}(\mathbf{S}^{-1} \odot \mathrm{diag}(\boldsymbol{\sigma}^2))_{ij} = \sum_i \sigma_i^2(\mathbf{S}^{-1})_{ii}$, this yields

$$\frac{\partial \mathrm{KL}\left[q(\boldsymbol{w}) \parallel p(\boldsymbol{w}|\mathbf{X}, \mathbf{y})\right]}{\partial \sigma_i^2} = \frac{1}{2}\frac{\partial \sigma_i^2(\mathbf{S}^{-1})_{ii}}{\partial \sigma_i^2} - \frac{1}{2}\frac{\partial \log \sigma_i^2}{\partial \sigma_i^2} = 0 \tag{3.7}$$

This results in $(\sigma_i^2)^{-1} = (\mathbf{S}^{-1})_{ii}$. This approximation has the effect of underestimating the variance for each variable (Murphy, 2012). In case we consider $\mathrm{KL}\left[p(\boldsymbol{w}|\mathbf{X}, \mathbf{y}) \parallel q(\boldsymbol{w})\right]$, the expression for the mean would remain the same but the variances would become $\sigma_i^2 = \mathbf{S}_{ii}$, which is the simplest way to approximate the correlated posterior over $\boldsymbol{w}$ but it is going to inflate the variance in case of strong correlations. Similar results can be also obtained for different posterior distributions, such as Gaussian posteriors with full or low-rank covariance, or matrix-variate Gaussian posteriors (Louizos and Welling, 2016).

3.2.3 Initialization for classification and convolutional layers

In this section we show how our proposal can be extended to k-class classification problems. We assume a one-hot encoding of the labels, so that $\mathbf{Y}$ is an $n \times k$ matrix of zeros and ones (one for each row of $\mathbf{Y}$). Recently, it has been shown that it is possible to obtain an accurate modeling of the posterior over classification functions by applying regression on a transformation of the labels (Milios, Camoriano, et al., 2018). This is interesting because it allows us to apply Bayesian linear regression as before in order to initialize VI for DNNs. The transformation of the labels is based on the formalization of a simple intuition, which is the inversion of the softmax transformation. One-hot encoded labels are viewed as a set of parameters of a degenerate Dirichlet distribution. We resolve the degeneracy of the Dirichlet distribution by adding a small regularization, say $\alpha = 0.01$, to the parameters. At this point, we leverage the fact that Dirichlet distributed random variables can be constructed as a ratio of Gamma random variables, that is, if $x_i \sim \text{Gamma}(a_i, b)$, then $\frac{x_i}{\sum_j x_j} \sim \text{Dir}(\mathbf{a})$. We can then approximate the Gamma random variables with log-Normals by moment matching, which become Gaussian after a logarithm transformation. By doing so, we obtain a representation of the labels which allows us to use standard regression with a Gaussian likelihood, and which retrieves an approximate Dirichlet when mapping predictions back using the softmax transformation. As a result, the latent functions obtained represent probabilities of class labels. The only small complication is that the transformation imposes a different noise level for labels that are 0 or 1, and this is due to the non-symmetric nature of the transformation. Nevertheless, it is a simple matter to extend Bayesian linear regression to handle heteroscedasticity; see the supplementary material and Milios, Camoriano, et al. (2018) for more insights on the transformation to apply regression on classification problems.

IBLM can also be applied to CNNs. Convolutional layers are commonly implemented as matrix multiplication (e.g. as a linear model) between a batched matrix of patches and a reshaped filter matrix (Jia, 2014). Rather than using the outputs of the previous layer as they are, for convolutional layers each Bayesian linear model learns the mapping from spatial patches to output features. In Algorithm 2 we sum-

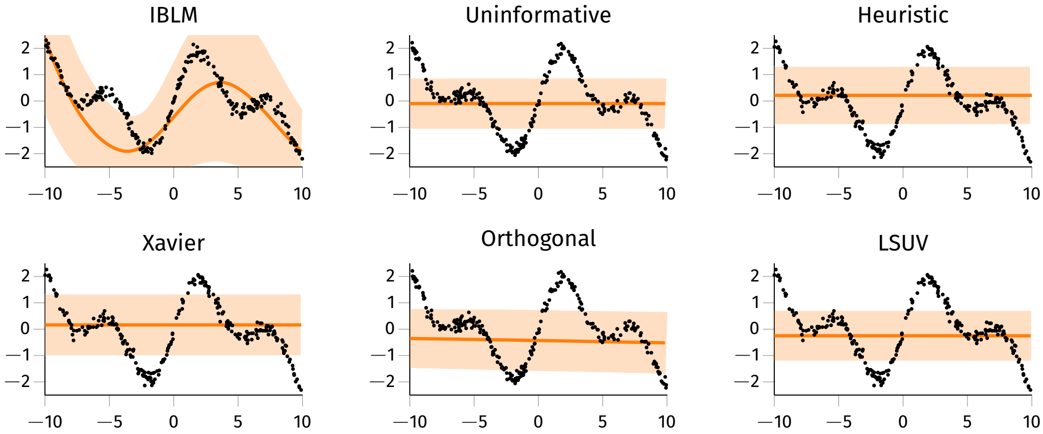


Figure 3.3: Illustration of the different initialization for the variational parameters in a simple 1D regression task.

marize a sketch of the proposed method for regression as well as for classification and convolutional layers.

Algorithm 2: Sketch of IBLM

```
Inputs  : Model $M$, Dataset $\mathcal{D}$
Returns : Initialized model
foreach layer in $M$ do
    foreach outfeature in layer do
        $X, Y \leftarrow$ next batch in $\mathcal{D}$;
        propagate $X$;
        $X_{\mathtt{IBLM}} \leftarrow$ output of previous layer;
        if layer is convolutional then
            $X_{\mathtt{IBLM}} \leftarrow$ patch extraction($X_{\mathtt{IBLM}}$);
        if likelihood is classification then
            $\mathrm{var}(Y_{\mathtt{IBLM}}) \leftarrow \log\left[(Y+\alpha)^{-1}+1\right]$;
            $\mathrm{mean}(Y_{\mathtt{IBLM}}) \leftarrow \log(Y+\alpha) - \mathrm{var}(Y_{\mathtt{IBLM}})/2$;
        else
            $Y_{\mathtt{IBLM}} \leftarrow Y$;
        Compute $p_{lm}(\boldsymbol{w} \mid \mathbf{X}, \mathbf{Y})$;
        Initialize $q(\boldsymbol{w})$ with the best approx. of $p_{lm}(\boldsymbol{w} \mid \mathbf{X}, \mathbf{Y})$
```

3.3 EXPERIMENTAL EVALUATION

In this section, we compare different initialization algorithms for variational inference to prove the effectiveness of IBLM. At layer (l), we choose priors $p(\mathbf{W}^{(l)}) =$

$\prod_{i,j} \mathcal{N}(w_{i,j}^{(l)} \,|\, 0, \frac{1}{D^{(l-1)}})$, where $D^{(l-1)}$ denotes the number of input features at layer (l), and focus on fully-factorized variational posteriors $q(\mathbf{W}^{(l)}) = \prod_{i,j} \mathcal{N}(w_{i,j}^{(l)} \,|\, \mu_{i,j}^{(l)}, (\sigma^2)_{i,j}^{(l)})$;

We propose a number of competitors inspired from the literature developed for loss minimization in DNNS and CNNS:

- **Uninformative.** The posteriors at each layer are initialized with zero mean and unit variance.

- **Random Heuristic.** An extension to commonly used heuristics with $\mu_{i,j}^{(l)} = 0$ and $(\sigma^2)_{i,j}^{(l)} = \frac{1}{D^{(l-1)}}$. Because this is the same as for the prior, this yields an initial KL divergence in the ELBO equal to zero.

- **Xavier Normal.** Originally proposed by Glorot and Bengio (2010), it samples all weights independently from a Gaussian distribution with zero mean and $(\sigma^2)_{i,j}^{(l)} = \frac{2}{D^{(l-1)}+D^{(l)}}$. This variance-based scaling avoids issues with vanishing or exploding gradients. We extend this to VI by directly setting $\mu_{i,j}^{(l)} = 0$ and $(\sigma^2)_{i,j}^{(l)} = \frac{2}{D^{(l-1)}+D^{(l)}}$, given that the sampling is performed during the Monte Carlo estimate of the log-likelihood.

- **Orthogonal.** Starting from the analysis of learning dynamics of DNNS with linear activations, Saxe et al. (2013) propose an initialization scheme with orthonormal weight matrices. The idea is to decompose a Gaussian random matrix onto an orthonormal basis, and use the resulting orthogonal matrix for initialization. We adapt this method to VI by initializing the mean matrix with the orthogonal matrix and $(\sigma^2)_{i,j}^{(l)} = \frac{1}{D^{(l-1)}}$. In the experiments, we make use of the Pytorch QR-decomposition (Paszke et al., 2019).

- **Layer-Sequential Unit-Variance (LSUV).** Starting from the orthogonal initialization, Mishkin and Matas (2016) propose a data-driven greedy layer-wise variance scaling of the weight matrices. We implement Layer-Sequential Unit-Variance (LSUV) for the means, while the variances are set to $(\sigma^2)_{i,j}^{(l)} = \frac{1}{D^{(l-1)}}$.

Figure 3.3 presents an illustration of the different initialization for the variational parameters in a simple 1D regression task. IBLM being data dependent already shows some visible fitting behaviour. We report an extensive validation of our proposal, with a series of experiments involving DNNS and CNNS for regression and classification, comparing it against other initializations for VI. For bigger models, like the CNNS, we will benchmark mean-field Gaussian variational inference with MCD (Gal and Ghahramani, 2016b) and Natural Noisy Gradients or NOISY-KFAC

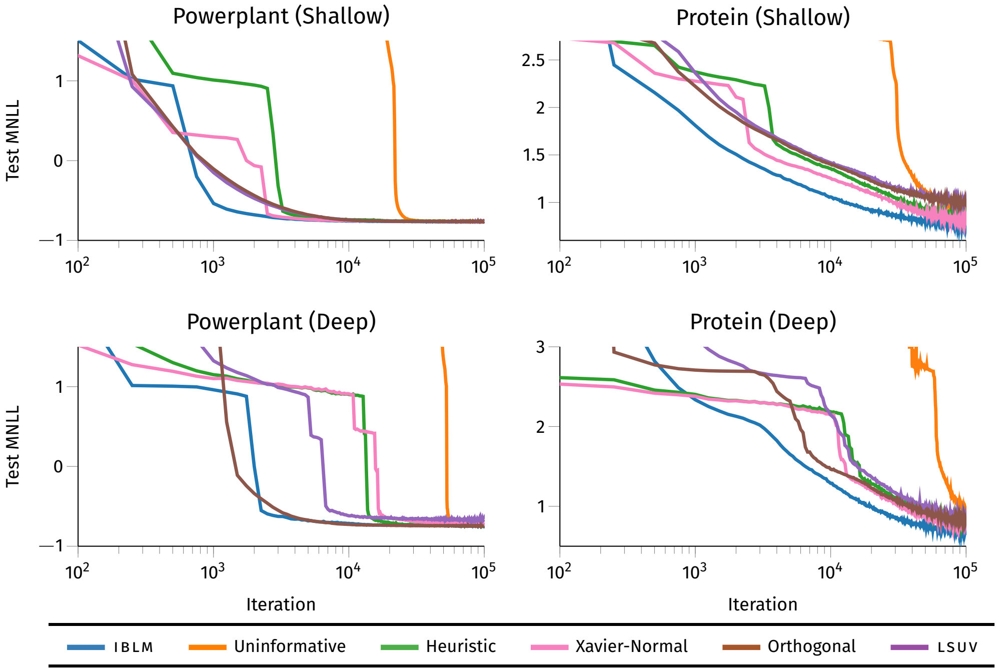


Figure 3.4: Progression of test mean negative loglikelihood (MNLL) with different initializations with shallow and deep architectures on Powerplant and Protein. Experiment repeated five times, only the average is shown.

(G. Zhang et al., 2018), which represent the two important references for inference of Bayesian CNNs. Throughout the experiments, we use ADAM (Kingma and Ba, 2015) as optimizer with learning rate of 10^{-3}, a batch-size of 64 examples and 16 Monte Carlo samples at training time and 128 at test time. All experiments are run on a server equipped with two 16c/32t Intel Xeon CPU and four NVIDIA Tesla P100, with a maximum time budget of 24 hours (never reached). To better understand the effectiveness of different initializations, all learning curves are plotted w.r.t. training iteration rather than wall-clock time.

3.3.1 The effect of initialization in deep variational neural networks

In this experiment we compare initialization methods for a shallow DNN architecture on two regression datasets, Powerplant ($N = 9568$, $D = 4$) and Protein ($N = 45730$, $D = 9$). The architecture used in these experiments has one single hidden layer with 100 hidden neurons and rectified linear unit (ReLU) activations. Figure 3.4 on the top row shows the learning curves repeated over five different train/test splits. First of all we observe that initialization hugely impact the convergence properties of variational inference. On Powerplant, for instance, the impact is easily quantified: between the fastest and the slowest configurations to converge there are differences as big as 30/40 times more training interations. This is an expected result and a trivial conclusion, but the effects of poor choices of initialization schemes in the convergence of variational inference were never discussed. Furthermore, IBLM allows for a better initialization compared to the competitors, leading to lower MNLL on the test set (curves for the root mean squared error (RMSE) are similar and available in the Appendix). Similar considerations hold when increasing the depth of the model, keeping the same experimental setup. Figure 3.4 shows what happens we we switch to a deeper architecture with five hidden layers and 100 hidden neurons per layer (ReLU activations).

3.3.2 Scaling up variational inference to deep convolutional neural networks

For this experiment, we implemented a Bayesian version of the original LeNet architecture proposed by LeCun, Bottou, et al. (1998) with two convolutional layers of 6 and 16 filters, respectively and ReLU activations applied after all convolutional layers and fully-connected layers. We tested our framework on MNIST and on CIFAR10 .

Before showing and discussing the comparisons, this is a challenging setup to validate two claims we made in the previous sections. Previously we claimed that (i) small batches of data are sufficient to solve the Bayesian linear model and that (ii) our initialization does not incur significant overheads. To justify such claims, we initialize a LeNet on MNIST with an increasing number of samples per batch; Figure 3.5 (on the left) shows how test MNLL just after the initialization is affected by this choice. Using the full training set leads to a better estimate of the posterior but from 64/128 samples the improvement on the test MNLL is only marginal. Note also that the mini-batch size affects also the heterogeneity of the posteriors, which vanishes when using the full training set. The same experiment is also repeated comparing test MNLL after initialization between VI with IBLM and MCD with the common Xavier initializer (Figure 3.5 on the right). Similar comments apply also for this case: IBLM allows the training to start from a marginally lower negative log-likelihood.

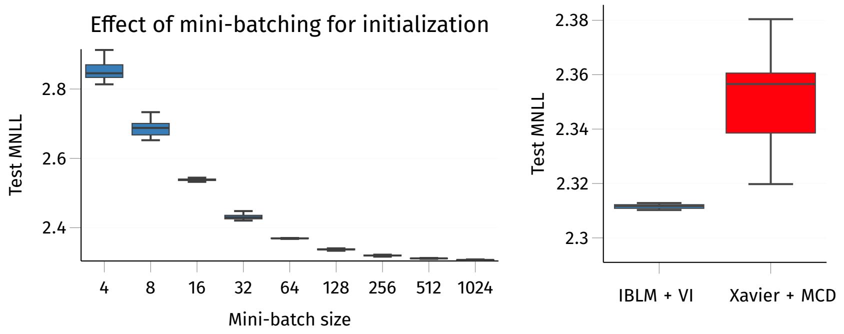


Figure 3.5: Comparison of test MNLL after initialization of LENET for MNIST averaged out of eight successive runs. On the left, IBLM with different batch sizes, on the right comparison with MCD.

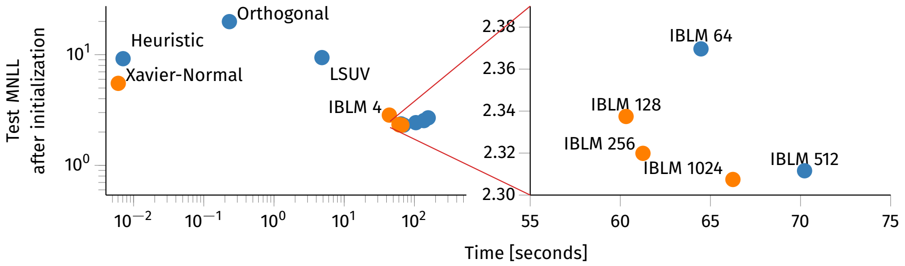


Figure 3.6: On the **left**, comparison of initialization time versus test MNLL, averaged out of eight successive runs (on the **right**, magnification of the small portion of the plot). Orange corresponds to the Pareto frontier. Before training, four out of five optimal initializers are IBLM.

Finally, Figure 3.6 reports the test MNLL after initialization as a function of the time required to initialize the model (orange points correspond to Pareto-optimal points). The timings and likelihood values are obtained by repeating the experiment eight time independently on a graphic processing unit (GPU) fully dedicated. The IBLM initializer takes roughly one minute to finish, which in the full timing budget of training such models is negligible.

Now we can move to the comparisons, to which we include a further initialization based on the maximum-a-posteriori (MAP). For this case we optimize the MAP loss with the same prior and we use this solution to initialize the $\mu^{(l)}_{i,j}$, while we set the variances to a small value of $\log[(\sigma^2)^{(l)}_{i,j}] = -5.5$ (for a fair comparison, we allow the MAP to train for the same amount of time required by IBLM to complete). Even for relatively small CNNs, we observe somethig alarming: the only initialization strategies that achieve convergence are the orthogonal and LSUV, along with IBLM and MAP, which are both data dependent; the other methods systematically make

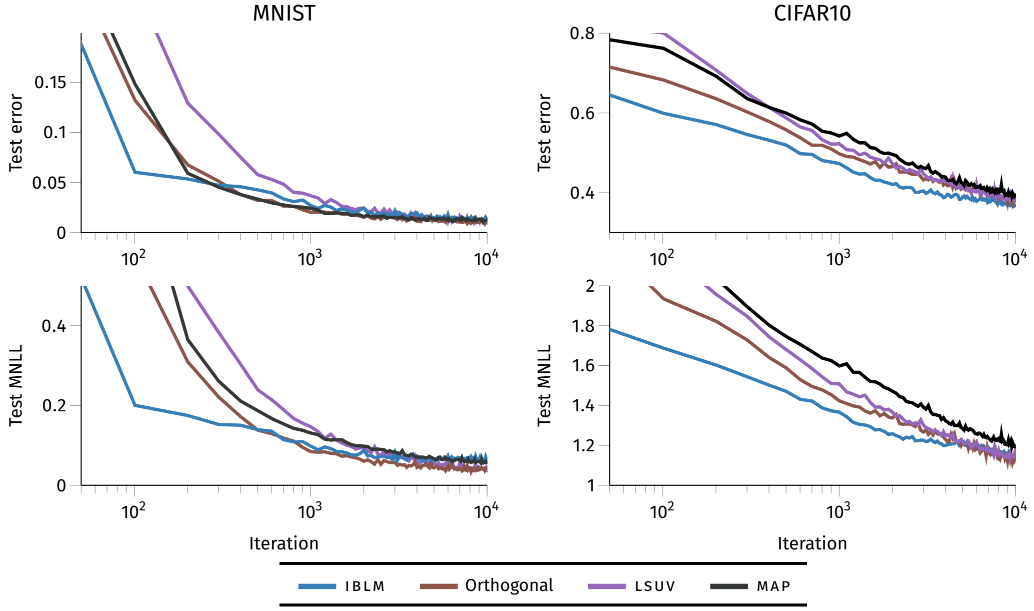


Figure 3.7: Progression of test error and test MNLL with different initializations on LeNet for MNIST and CIFAR10. Note that the Uniformative, Heuristic and Xavier intializers did not converge, and as such they are not included in the plots.

the optimization to not make progress, pushing the posterior back to the prior. **??** reports the progression of the error rate and MNLL. For both MNIST and CIFAR10 , IBLM places the parameters where the network can consistently deliver better performance both in terms of error rate and MNLL throughout the entire learning procedure.

3.3.3 Comparison with variational inference beyond mean field Gaussian

Monte Carlo Dropout (Gal and Ghahramani, 2016a) offers a simple and effective way to perform approximate Bayesian CNN inference, thanks to the connection between dropout (Simonyan and Zisserman, 2014) and variational inference. In this experiment, we aim to compare and discuss benefits and disadvantages of using a Gaussian posterior approximation with respect to the Bernoulli approximation that characterizes MCDs. For a fair comparison, we implemented the same LENET architecture and the same learning procedure as in Gal and Ghahramani (2016a)[1], with the two convolutional layers having 20 and 50 filters, respectively. We also use the same learning rate policy $\lambda \times (1 + \tau \times \mathtt{iter})^{-p}$ with $\tau = 0.0001$, $p = 0.75$,

---

[1] https://github.com/yaringal/DropoutUncertaintyCaffeModels

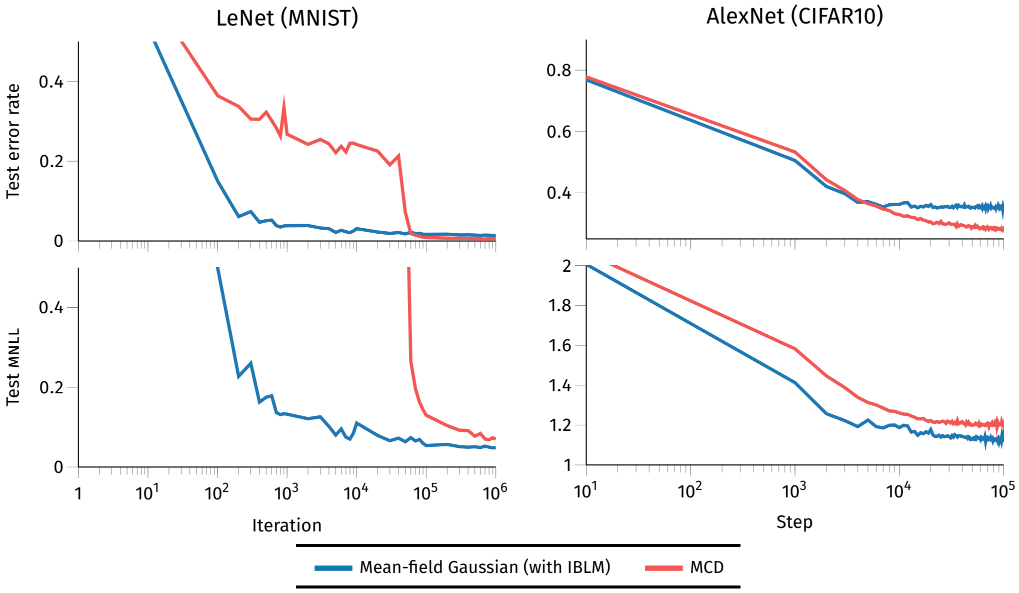


Figure 3.8: Progression of test error and MNLL for two different convolutional neural networks on both MNIST and CIFAR10. Variational inference is initialized with IBLM.

$\lambda = 0.01$ and weight decay of 0.0005. Figure 3.8 on the left shows the learning curves: MCD achieves lower error rate but its rough approximation imposed by the Bernoulli-like posterior is reflected on an higher MNLL compared to VI with a Gaussian posterior. Provided with a good initialization, Gaussian VI can better fit the model and deliver a better quantification of uncertainty.

Similar comments apply also for CIFAR10 on ALEXNET (Krizhevsky, Sutskever, et al., 2012), a CNN composed by a stack of five convolutional layers and three fully-connected layers for a total of more than 1M parameters (2M for VI) (results in Figure 3.8 on the right). Note that in this case, the test error seems to favor MCD but the mean-field Gaussian approximation deliver better test likelihood (1.15 vs 1.38 for MCD). Calibration of uncertainty is an important performance metric that one should take into account for comparing classification models (Flach, 2016; Guo et al., 2017). Reliability Diagrams and the Expected Calibration Error (ECE) are standard methods to empirically estimate the calibration uncertainty. Reliability Diagrams are a visualization tool where sample accuracy is plotted as function of confidence (DeGroot and Fienberg, 1983; Niculescu-Mizil and Caruana, 2005). For a perfectly calibrated model, the diagram follows the identity function. Expected

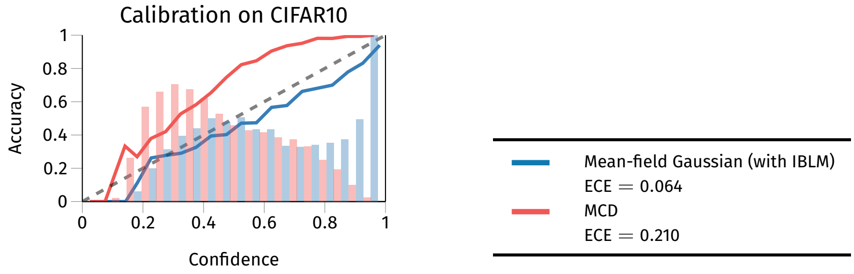
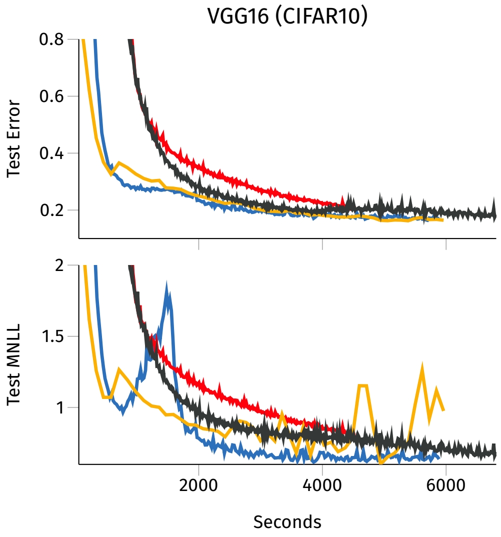


Figure 3.9: Reliability diagram and the expected calibration error for AlexNet trained on CIFAR10.



	Test MNLL	Test error
🟦 Gaussian VI with IBLM	0.637	0.167
⬛ Gaussian VI with MAP init.	0.750	0.201
🟥 MCD	0.821	0.215
🟨 NNG (Noisy-KFAC)	0.750*	0.164

Note: Due to high variance in the test MNLL for the noisy natural gradients, we report the minimum value of the last 5 epochs.

Figure 3.10: Progression of test error and MNLL for a very deep convolutional neural network trained on CIFAR10.

Calibration Error represents a summary statistic of the calibration (Naeini et al., 2015) and it is defined as follows,

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right| , \tag{3.8}$$

where the test set is divided into M disjoint subsets $\{B_1, \dots, B_M\}$, each of them corresponding to a given interval of the predicted confidence. Figure 4.9 shows the reliability diagrams and the ECE for AlexNet trained on CIFAR10.

Finally, we demonstrate that – provided with a sensible initialization – even simple factorized Gaussian posterior can achieve state-of-the-art performance on CIFAR10 with VGG [2], a large scale CNN (Simonyan and Zisserman, 2014). In this experiment, in addition to MCD, we also compare with NOISY-KFAC, an approximation of matrix-variate Gaussian posterior using noisy natural gradients introduced by G. Zhang et al. (2018). The four models are trained with a maximum time budget of 100 minutes for the entire end-to-end training (curves are shifted by the initialization time). The Results are shown in Figure 3.10 and in the adjacent table. In this experiment, we have experienced the situation in which, the ELBO is completely dominated by the KL divergence. The KL regularization term in the variational objective severely penalizes training of over-parameterized model. With a sensible initialization of this kind of model, the approximate posterior is drastically different from a spherical Gaussian prior and the variational objective is majorly dominated by the regularization term rather than the reconstruction likelihood. To deal with such issue, we propose and implement a simple policy to gradually include the KL term in the ELBO. Given the generic expression for the ELBO, we modify the lower bound as follow:

---

[2] We implemented the same architecture as in G. Zhang et al. (2018)

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\nu}) = \mathbb{E}_{q(\theta;\boldsymbol{\nu})} \log p(\mathbf{y} \mid \theta) - \lambda \mathrm{KL}\left[q(\theta;\boldsymbol{\nu}) \parallel p(\theta)\right]$$

with $\lambda = \gamma\,(1 + \exp(-\alpha(\text{iter} - \beta)))^{-1}$. This way, we start the optimization of the ELBO with a low regularization, and progressively increase it throughout the optimization. Effectively, this is equivalent to start with a prior with very weak regularization strength and slowly moving towards a proper spherical Gaussian. For the experiment on VGG16, we used $\alpha = 2 \cdot 10^{-3}$, $\beta = 2.5 \cdot 10^{4}$ and $\gamma = 10^{-1}$. The practice of down-playing the KL term was initially introduced for generative models like variational autoencoders (VAEs) (Bowman et al., 2016; Sønderby et al., 2016), but now accepted by the Bayesian deep learning community as a quick remedy for poor performance.

3.4 FINAL REMARKS

Before moving to some final remarks, let's quick summarize the empirical evidence up to this point. We prove that the role of initialization is severely under reported in the literature of variational inference for Bayesian neural network. The reasons for the failing of Bayesian neural networks (BNNs) in practice might depends heavily on poor scheme for the initialization of the variational parameters. The test likelihood at the beginning of the optimization seems to be a good indication of a possible failure in the training. See for example the situation in Figure 3.11 (LeNet

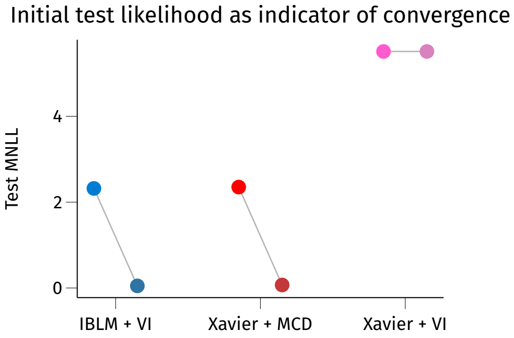


Figure 3.11: Test likelihood after initialization (on the left) and at the end of the training procedure (on the right) for LeNet on MNIST.

on MNIST): both variational inference with IBLM and MCD with Xavier have comparable initial test likelihood and both converge while variational inference with Xavier starts with higher test likelihood and fails to progress. It's unclear whether this strong conjecture holds consistently, but this evidence shows that a very high initial test likelihood might be a symptom of bad conditioning. This figure also highlights a pathology which might be intrinsic with the nature of mean-field Gaussian posterior: with the same initialized network, Gaussian VI fails to make any progress during training, while MCD converges fairly easily. This motivates further research on non-Gaussian posteriors or, at least, the departure from factorized distributions.

This work fills an important gap in the literature of Bayesian deep learning, that is how to effectively initialize variational parameters in VI. We proposed a novel way to do so, IBLM, which is based on an iterative layer-wise initialization based on Bayesian linear models. Through a series of experiments, including regression and classification with DNNS and CNNS, we demonstrated the ability of our approach to consistently initialize the optimization in a way that makes convergence faster than alternatives inspired from the state-of-the-art in loss minimization for deep learning.

4

EFFICIENT PARAMETERIZATIONS FOR VARIATIONAL POSTERIORS

Over-parameterized models, such as deep neural networks and convolutional neural networks, form a class of models that are routinely adopted in a wide variety of applications, and for which Bayesian inference is desirable but extremely challenging. Variational inference offers the tools to tackle this challenge in a scalable way and with some degree of flexibility on the approximation, but for over-parameterized models this is challenging due to the over-regularization property of the variational objective. Inspired by the literature on kernel methods, and in particular on structured approximations of distributions of random matrices, this chapter will discuss Walsh-Hadamard variational inference (WHVI), a new parameterization for variational inference which leverages Walsh-Hadamard-based factorization strategies to reduce the parameterization and accelerate computations. Extensive theoretical and empirical analyses demonstrate that WHVI yields considerable speedups and model reductions compared to other techniques to carry out approximate inference for over-parameterized models, and ultimately show how advances in kernel methods can be translated into advances in approximate Bayesian inference for Deep Learning.

4.1 THE PROBLEM OF OVERPARAMETERIZATION IN VARIATIONAL INFERENCE

Since its inception, variational inference (VI) (Jordan et al., 1999) has continuously gained popularity as a scalable and flexible approximate inference scheme for a variety of models for which exact Bayesian inference is intractable. Bayesian neural networks (Mackay, 1994; Neal, 1996) represent a good example of models for which inference is intractable, and for which VI – and approximate inference in general – is challenging due to the nontrivial form of the posterior distribution and the large dimensionality of the parameter space (Graves, 2011; Gal and Ghahramani, 2016b). Recent advances in VI allow one to effectively deal with these issues in various ways. For instance, a flexible class of posterior approximations can be constructed using, e.g., normalizing flows (D. Rezende and Mohamed, 2015), whereas the need to operate with large parameter spaces has pushed the research in the direction of Bayesian compression (Louizos, Ullrich, et al., 2017; Molchanov et al., 2017).

We saw in the previous chapters that employing VI is notoriously challenging for Bayesian deep learning, and even more so for over-parameterized statistical models. Let's consider a supervised learning task with N input vectors and corresponding labels collected in $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathbf{y} = \{y_1, \dots, y_N\}$, respectively; furthermore, let's consider deep neural networks (DNNs) with weight matrices $\theta = \left\{\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}\right\}$, likelihood $p(\mathbf{y} \mid \theta, \mathbf{X})$, and prior $p(\theta)$. Following standard variational arguments, after introducing an approximation $q(\theta; \boldsymbol{\nu})$ to the posterior $p(\mathbf{W} \mid \mathbf{y}, \mathbf{X})$ it is possible to obtain a lower bound to the log-marginal likelihood $\log p(\mathbf{y} \mid \mathbf{X})$ as follows:

$$\log p(\mathbf{y} \mid \mathbf{X}) \geqslant \mathbb{E}_{q(\theta;\boldsymbol{\nu})} \log p(\mathbf{y} \mid \theta, \mathbf{X}) - \mathrm{KL}\left[q(\theta; \boldsymbol{\nu}) \parallel p(\theta)\right] . \tag{4.1}$$

The first term acts as a model fitting term, whereas the second one acts as a regularizer, penalizing solutions where the posterior is far away from the prior. It is easy to verify that the Kullback-Leibler (KL) term can be the dominant one in the objective for over-parameterized models. For example, a mean field posterior approximation turns the KL term into a sum of as many KL terms as the number of model parameters, say Q, which can dominate the overall objective when $Q \gg N$. As a result, the optimization focuses on keeping the approximate posterior close to the prior, disregarding the rather important model fitting term. This issue has been observed in a variety of deep models (Bowman et al., 2016), where it was proposed to gradually include the KL term throughout the optimization (Bowman et al., 2016; Sønderby et al., 2016) to scale up the model fitting term (A. G. Wilson and Izmailov, 2020a; Wenzel et al., 2020) or to improve the initialization of variational parameters (Rossi, Michiardi, et al., 2019). Alternatively, other approximate inference methods for deep models with connections to VI have been proposed, notably Monte Carlo

Dropout (Monte Carlo dropout (MCD); Gal and Ghahramani, 2016b) and Noisy Natural Gradients (G. Zhang et al., 2018).

4.1.1 Contributions

In this chapter we make the several contributions. We propose a novel strategy to cope with model over-parameterization when using variational inference, which is inspired by the literature on kernel methods. Our proposal is to reparameterize the variational posterior over model parameters by means of a structured decomposition based on random matrix theory (Tropp, 2011), which has inspired a number of fundamental contributions in the literature on approximations for kernel methods, such as Fastfood (Le et al., 2013) and Orthogonal Random Features (ortogonal random features (ORF), (F. X. Yu et al., 2016)). The key operation within our proposal is the Walsh-Hadamard transform, and this is why we name our proposal Walsh-Hadamard Variational Inference (WHVI).

Without loss of generality, consider Bayesian DNNs with weight matrices $W^{(l)}$ of size $D \times D$. Compared with mean field VI, WHVI has a number of attractive properties. The number of parameters is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D)$, thus reducing the over-regularization effect of the KL term in the variational objective. We derive expressions for the reparameterization and the local reparameterization tricks, showing that, the computational complexity is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$. Finally, unlike mean field VI, WHVI induces a matrix-variate distribution to approximate the posterior over the weights, thus increasing flexibility at a log-linear cost in D instead of linear.

We can think of our proposal as a specific factorization of the weight matrix, so we can speculate that other tensor factorizations (Novikov et al., 2015) of the weight matrix could equally yield such benefits. Our comparison against various matrix factorization alternatives, however, shows that WHVI is superior to other parameterizations that have the same complexity. Furthermore, while matrix-variate posterior approximations have been proposed in the literature of VI (Louizos and Welling, 2016), this comes at the expense of increasing the complexity, while our proposal keeps the complexity to log-linear in D.

Through a wide range of experiments on DNNs and convolutional neural networks (CNNs), we demonstrate that our approach enables the possibility to run variational inference on complex over-parameterized models, while being competitive with state-of-the-art alternatives. Ultimately, our proposal shows how advances in kernel methods can be instrumental in improving VI, much like previous works showed how kernel methods can improve, e.g., Markov chain Monte Carlo sampling (Sejdinovic et al., 2014; Strathmann et al., 2015) and statistical testing (Gretton, Fukumizu, et al., 2008; Gretton, Borgwardt, et al., 2012; Zaremba et al., 2013).

4.2 STRUCTURED APPROXIMATIONS FOR KERNEL MATRICES

WHVI is inspired by a line of works that developed from random feature expansions for kernel machines (Rahimi and Recht, 2008), which we briefly review here. A positive-definite kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ induces a mapping $\boldsymbol{\phi}(\mathbf{x})$, which can be infinite dimensional depending on the choice of $\kappa(\cdot, \cdot)$. Among the large literature of scalable kernel machines, random feature expansion techniques aim at constructing a finite approximation to $\boldsymbol{\phi}(\cdot)$. Denoting by $\mathbf{K}$ the matrix with elements $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ and by $\boldsymbol{\Phi}$ the matrix with rows $\boldsymbol{\phi}(\mathbf{x}_i)^\top$, this approximation yields $\mathbf{K} \approx \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ and this allows one to turn kernel machines into linear models in this kernel-induced representation $\boldsymbol{\Phi}$. For many kernel functions (Rahimi and Recht, 2008; Cho and Saul, 2009), this approximation is built by applying a nonlinear transformation to a random projection $\mathbf{X}\boldsymbol{\Omega}$, where $\boldsymbol{\Omega}$ has entries $\mathcal{N}(\omega_{ij}|0, 1)$. If the matrix of training points $\mathbf{X}$ is $N \times D$ and we are aiming to construct D random features, that is $\boldsymbol{\Omega}$ is $D \times D$, this requires N times $\mathcal{O}(D^2)$ time, which can be prohibitive when D is large.

Fastfood (Le et al., 2013) tackles the issue of large dimensional problems by replacing the matrix $\boldsymbol{\Omega}$ with a random matrix for which the space complexity is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D)$ and time complexity of performing products with input vectors is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$. In Fastfood, the matrix $\boldsymbol{\Omega}$ is replaced by

$$\boldsymbol{\Omega} \approx \mathbf{S}\mathbf{H}\mathbf{G}\boldsymbol{\Pi}\mathbf{H}\mathbf{B}, \tag{4.2}$$

where $\boldsymbol{\Pi}$ is a permutation matrix, $\mathbf{H}$ is the Walsh-Hadamard matrix, whereas $\mathbf{G}$ and $\mathbf{B}$ are diagonal random matrices with standard Normal and Rademacher ($\{\pm 1\}$)

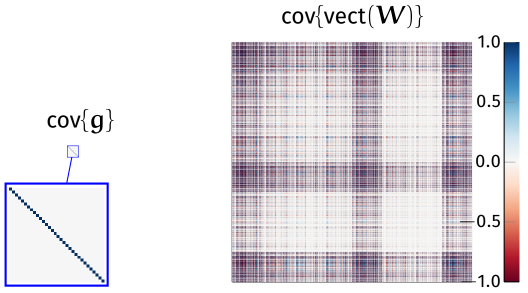


Figure 4.1: Normalized covariance of $\mathbf{g}$ and $\text{vect}(\boldsymbol{W})$.

distributions, respectively. The Walsh-Hadamard matrix is defined recursively starting from

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \text{and then} \quad H_{2D} = \begin{bmatrix} H_D & H_D \\ H_D & -H_D \end{bmatrix}, \tag{4.3}$$

which it could possibly be scaled by $D^{-1/2}$ to make it orthonormal. The product $\mathbf{Hx}$ can be computed in $\mathcal{O}(D \log D)$ time and $\mathcal{O}(1)$ space using the in-place version of the Fast Walsh-Hadamard Transform (FWHT, Fino and Algazi, 1976). $\boldsymbol{S}$ is also diagonal with i.i.d. entries, and it is chosen such that the elements of $\boldsymbol{\Omega}$ obtained by this series of operations are approximately independent and follow a standard Normal (see (Tropp, 2011) for more details). Fastfood inspired a series of other works on kernel approximations , whereby Gaussian random matrices are approximated by a series of products between diagonal Rademacher and Walsh-Hadamard matrices (F. X. Yu et al., 2016; Bojarski et al., 2017).

4.3 FROM STRUCTURED KERNEL APPROXIMATIONS TO WALSH-HADAMARD VARIATIONAL INFERENCE

Fastfood and its variants yield cheap approximations to Gaussian random matrices with pseudo-independent entries, and zero mean and unit variance. The question we address here is whether we can use these types of approximations as cheap approximating distributions for VI. Let's start by considering a prior for the elements of the diagonal matrix $\mathbf{G} = \text{diag}(\mathbf{g})$ and a variational posterior $q(\mathbf{g}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ of Equation (4.2). From here we can actually obtain a class of approximate posterior with some desirable properties as discussed next. Let $\mathbf{W} = \mathbf{W}^{(l)} \in \mathbb{R}^{D \times D}$ be the weight matrix of a DNN at layer (l), and consider

$$\widetilde{\mathbf{W}} \sim q(\mathbf{W}) \quad \text{s.t.} \quad \widetilde{\mathbf{W}} = \mathbf{S}_1 \mathbf{H} \operatorname{diag}(\tilde{\mathbf{g}}) \mathbf{H} \mathbf{S}_2 \quad \text{with} \quad \tilde{\mathbf{g}} \sim q(\mathbf{g}). \tag{4.4}$$

4.3.1 Statistical properties of the structure induced by WHVI

The choice of a Gaussian $q(\mathbf{g})$ and the linearity of the operations in Equation (4.4) induce a parameterization of a matrix-variate Gaussian distribution for $\mathbf{W}$, which is controlled by $\mathbf{S}_1$ and $\mathbf{S}_2$ if we assume that we can optimize these diagonal matrices. For a generic $D_1 \times D_2$ matrix-variate Gaussian distribution, we have

$$\mathbf{W} \sim \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V}) \quad \text{if and only if} \quad \operatorname{vect}(\mathbf{W}) \sim \mathcal{N}(\operatorname{vect}(\mathbf{M}), \mathbf{V} \otimes \mathbf{U}), \tag{4.5}$$

where $\mathbf{M} \in \mathbb{R}^{D_1 \times D_2}$ is the mean matrix and $\mathbf{U} \in \mathbb{R}^{D_1 \times D_1}$ and $\mathbf{V} \in \mathbb{R}^{D_2 \times D_2}$ are two positive definite covariance matrices among rows and columns, and $\otimes$ denotes the Kronecker product. Matrix-variate Gaussian posteriors for variational inference have been introduced in Louizos and Welling (2016); however, assuming full covariance matrices $\mathbf{U}$ and $\mathbf{V}$ is memory and computationally intensive (quadratic and cubic in D, respectively). WHVI captures covariances across weights (see Figure 4.1), while keeping memory requirements linear in D and complexity log-linear in D, and we shall now see why.

In WHVI, as $\mathbf{S}_2$ is diagonal, $\mathbf{H}\mathbf{S}_2 = [\mathbf{v}_1, \dots, \mathbf{v}_D]$ with $\mathbf{v}_i = (\mathbf{S}_2)_{i,i}(\mathbf{H})_{:,i}$, so $\mathbf{W}$ can be rewritten in terms of $\mathbf{A} \in \mathbb{R}^{D^2 \times D}$ and $\mathbf{g}$ as follows

$$\operatorname{vect}(\mathbf{W}) = \mathbf{A}\mathbf{g} \quad \text{where} \quad \mathbf{A} = \begin{bmatrix} \mathbf{S}\mathbf{H}\operatorname{diag}(\mathbf{v}_0) \\ \vdots \\ \mathbf{S}\mathbf{H}\operatorname{diag}(\mathbf{v}_{d-1}) \end{bmatrix} \tag{4.6}$$

This rewriting, shows that the choice of $q(\mathbf{g})$ yields $q(\operatorname{vect}(\mathbf{W})) = \mathcal{N}(\mathbf{A}\boldsymbol{\mu}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$, proving that WHVI assumes a matrix-variate distribution $q(\mathbf{W})$, see Figure 4.1 for an illustration of this.

We now derive the parameters of the matrix-variate distribution $q(\mathbf{W}) = \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$ of the weight matrix $\tilde{\mathbf{W}} \in \mathbb{H}^{D \times D}$ given by WHVI,

$$\tilde{\mathbf{W}} = \mathbf{S}_1 \mathbf{H} \operatorname{diag}(\tilde{\mathbf{g}}) \mathbf{H} \mathbf{S}_2 \quad \text{with} \quad \tilde{\mathbf{g}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{4.7}$$

The mean $\mathbf{M} = \mathbf{S}_1\mathbf{H}\mathrm{diag}(\boldsymbol{\mu})\mathbf{H}\mathbf{S}_2$ derives from the linearity of the expectation. The covariance matrices $\mathbf{U}$ and $\mathbf{V}$ are non-identifiable: for any scale factor $s > 0$, we have $\mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$ equals $\mathcal{MN}(\mathbf{M}, s\mathbf{U}, \frac{1}{s}\mathbf{V})$. Therefore, we constrain the parameters such that $\mathrm{Tr}(\mathbf{V}) = 1$. The covariance matrices verify (see e.g. Section 1 in the supplement of Ding and Cook, 2014)

$$\mathbf{U} = \mathbb{E}_{q(\mathbf{W})}\left[(\mathbf{W}-\mathbf{M})(\mathbf{W}-\mathbf{M})^\top\right]$$
$$\mathbf{V} = \frac{1}{\mathrm{Tr}(\mathbf{U})}\mathbb{E}_{q(\mathbf{W})}\left[(\mathbf{W}-\mathbf{M})^\top(\mathbf{W}-\mathbf{M})\right].$$

Denoting by $\boldsymbol{\Sigma}^{1/2}$ a root of $\boldsymbol{\Sigma}$ and considering $p(\boldsymbol{\varepsilon}) = \mathcal{N}(0, \mathbf{I})$, we have

$$\mathbf{U} = \mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\mathbf{S}_1\mathbf{H}\mathrm{diag}(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\varepsilon})\mathbf{H}\mathbf{S}_2^2\mathbf{H}\mathrm{diag}(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\varepsilon})\mathbf{H}\mathbf{S}_1\right]. \tag{4.8}$$

Note that the Walsh-Hadamard matrix $\mathbf{H}$ *is symmetric, while* $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *are diagonal.*

Let's define the matrix $\mathbf{T}_2 \in \mathbb{H}^{D\times D^2}$ where the i^{th} row is the column-wise vectorization of the matrix $(\boldsymbol{\Sigma}_{i,j}^{1/2}(\mathbf{H}\mathbf{S}_2)_{i,j'})_{j,j'\leqslant D}$,

$$\mathbf{T}_1 = \begin{bmatrix} \mathrm{vect}\left(\boldsymbol{\Sigma}_{1,:}(\mathbf{H}\mathbf{S}_2)_{1,:}^\top\right)^\top \\ \vdots \\ \mathrm{vect}\left(\boldsymbol{\Sigma}_{D,:}(\mathbf{H}\mathbf{S}_2)_{d,:}^\top\right)^\top \end{bmatrix}. \tag{4.9}$$

We have that

$$\begin{aligned}
(\mathbf{T}_2\mathbf{T}_2^\top)_{i,i'} &= \sum_{j,j'=1}^{D} \boldsymbol{\Sigma}_{i,j}^{1/2}\boldsymbol{\Sigma}_{i',j}^{1/2}(\mathbf{H}\mathbf{S}_2)_{i,j'}(\mathbf{H}\mathbf{S}_2)_{i',j'} \\
&= \sum_{j,j',j''=1}^{D} \boldsymbol{\Sigma}_{i,j}^{1/2}(\mathbf{H}\mathbf{S}_2)_{i,j'}\mathbb{E}_{p(\boldsymbol{\varepsilon})}[\epsilon_j\epsilon_{j''}]\boldsymbol{\Sigma}_{i',j''}^{1/2}(\mathbf{H}\mathbf{S}_2)_{i',j'} \\
&= \sum_{j'=1}^{D}\mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\left(\sum_{j=1}^{D}\epsilon_j\boldsymbol{\Sigma}_{i,j}^{1/2}(\mathbf{H}\mathbf{S}_2)_{i,j'}\right)\left(\sum_{j''=1}^{D}\epsilon_{j''}\boldsymbol{\Sigma}_{i',j''}^{1/2}(\mathbf{H}\mathbf{S}_2)_{i',j'}\right)\right] \\
&= \mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\left(\mathrm{diag}(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\varepsilon})\mathbf{H}\mathbf{S}_2^2\mathbf{H}\mathrm{diag}(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\varepsilon})\right)_{i,i'}\right].
\end{aligned}$$

Using Equation (4.8), a root of $\mathbf{U} = \mathbf{U}^{1/2}\mathbf{U}^{1/2^\top}$ can be found:

$$\mathbf{U}^{1/2} = \mathbf{S}_1\mathbf{H}\mathbf{T}_2. \tag{4.10}$$

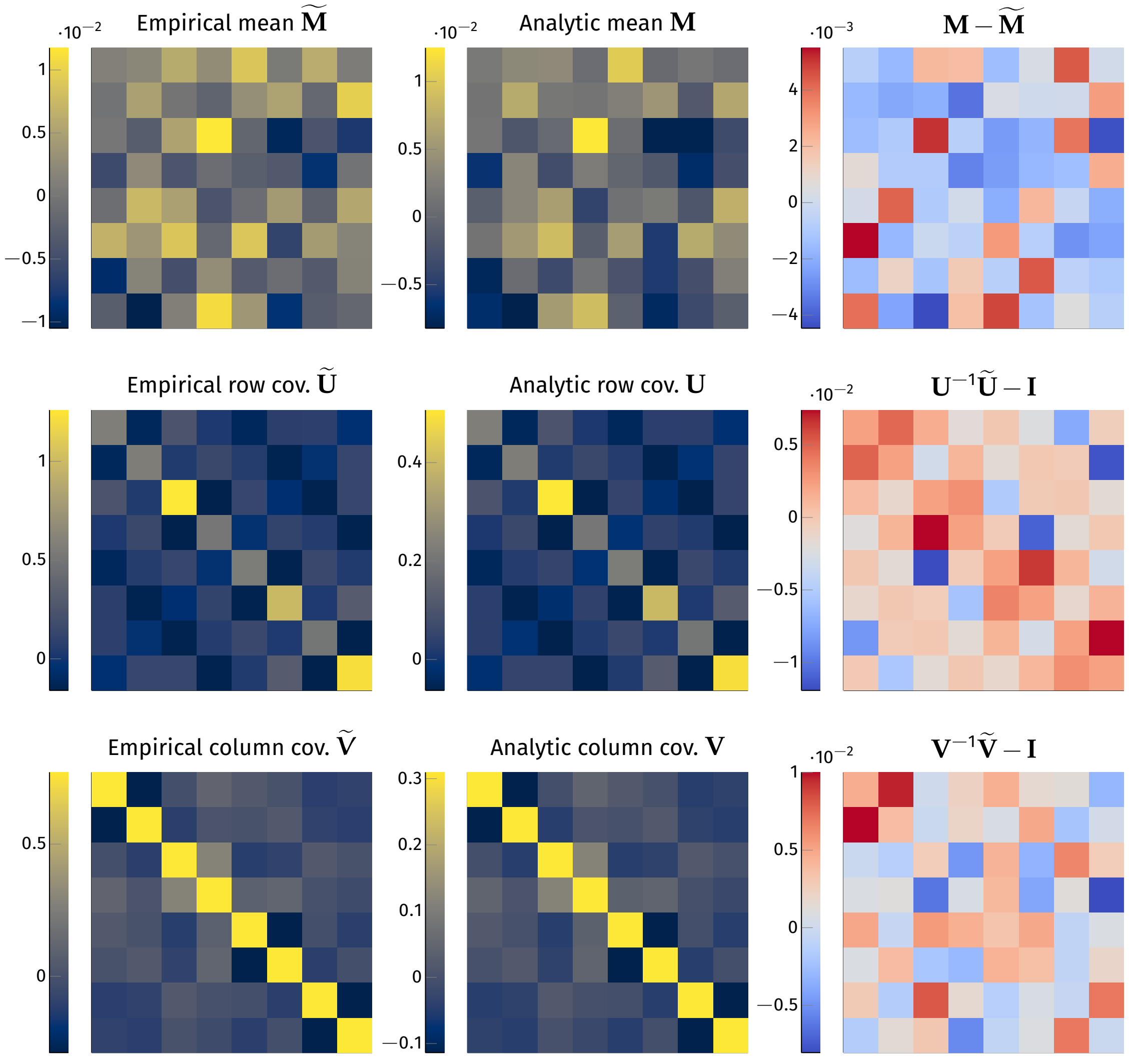


Figure 4.2: Numerical verification of first and second moments for WHVI for one choice of S_1, S_2 and $q(\mathbf{g})$. The empirical quantities are obtained with Monte Carlo samples.

Similarly for $\mathbf{V}$, we have

$$\mathbf{V}^{1/2} = \frac{1}{\sqrt{\mathrm{Tr}(\mathbf{U})}} S_2 H T_1, \quad \text{with} \quad T_1 = \begin{bmatrix} \mathrm{vect}\left(\Sigma_{1,:}\,(HS_1)^\top_{1,:}\right)^\top \\ \vdots \\ \mathrm{vect}\left(\Sigma_{D,:}\,(HS_1)^\top_{d,:}\right)^\top \end{bmatrix}. \tag{4.11}$$

The form of the moments for the matrix variate Gaussian are non-trivial and they can be confusing; to verify the correctness of these formulas we run a numerical study: we compute the mean and the two covariances analitically and empirically after sampling 15 000 times from Equation (4.4) and we report their deviations in Figure 4.2. For the three moments the errors are well within what we can consider acceptable and, hence we can validate the expressions we derived before.

GEOMETRICAL INTERPRETATION OF WHVI. The matrix $\mathbf{A}$ in Equation (4.6) expresses the linear relationship between the weights $\boldsymbol{W} = S_1 H \mathrm{diag}(\mathbf{g}) H S_2$ and the variational random vector $\mathbf{g}$, i.e. $\mathrm{vect}(\boldsymbol{W}) = \mathbf{A}\mathbf{g}$. Recall the definition of

$$\mathbf{A} = \begin{bmatrix} S_1 H \mathrm{diag}(\boldsymbol{\nu}_1) \\ \vdots \\ S_1 H \mathrm{diag}(\boldsymbol{\nu}_D) \end{bmatrix}, \quad \text{with } \boldsymbol{\nu}_i = (\mathbf{S}_2)_{i,i} (\mathbf{H})_{:,i}. \tag{4.12}$$

We show that a LQ-decomposition of $\mathbf{A}$ can be explicitly formulated as.

$$\begin{aligned} \mathrm{vect}(\boldsymbol{W}) &= [s_i^{(2)} S_1 H \mathrm{diag}(\mathbf{h}_i)]_{i=1,\dots,D}\ \mathbf{g} \\ &= \mathbf{L}\mathbf{Q}\mathbf{g}, \end{aligned} \tag{4.13}$$

where $\mathbf{h}_i$ is the i^{th} column of $\mathbf{H}$, $\mathbf{L} = \mathrm{diag}((s_i^{(2)} \mathbf{s})_{i=1,\dots,D})$, $\mathrm{diag}(\mathbf{s}^{(1)}) = \mathbf{S}_1$, $\mathrm{diag}(\mathbf{s}^{(2)}) = \mathbf{S}_2$, and $\mathbf{Q} = [\mathrm{H}\mathrm{diag}(\mathbf{h}_i)]_{i=1,\dots,D}$. Equation (4.13) derives directly from block matrix and vector operations. As $\mathbf{L}$ is clearly lower triangular (even diagonal), let us proof that $\mathbf{Q}$ has orthogonal columns. Defining the $d \times d$ matrix $\mathbf{Q}^{(i)} = \mathbf{H}\mathrm{diag}(\mathbf{h}_i)$, we have:

$$\mathbf{Q}^\top \mathbf{Q} = \sum_{i=1}^{D} \mathbf{Q}^{(i)\top} \mathbf{Q}^{(i)} = \sum_{i=1}^{D} \mathrm{diag}(\mathbf{h}_i) \mathbf{H}^\top \mathbf{H} \mathrm{diag}(\mathbf{h}_i) = \sum_{i=1}^{D} \mathrm{diag}(\mathbf{h}_i^2) = \sum_{i=1}^{D} D^{-1}\mathbf{I} = \mathbf{I}.$$

The mean of the structured matrix-variate posterior assumed by WHVI can span a D-dimensional linear subspace within the whole D^2-dimensional parameter space, and the orientation is controlled by the matrices $\mathbf{S}_1$ and $\mathbf{S}_2$. In more details, this decomposition gives direct insight on the role of the Walsh-Hadamard transforms: with complexity $D \log(D)$, they perform fast rotations $\mathbf{Q}$ of vectors living in a space of dimension D into a space of dimension D^2. Treated as parameters gathered in $\mathbf{L}$, $\mathbf{S}_1$ and $\mathbf{S}_2$ control the orientation of the subspace by distortion of the canonical axes.

4.3.2 Reparameterizations in WHVI for stochastic optimization

The so-called *reparameterization trick* (Kingma and Welling, 2014) is a standard way to make the variational lower bound in Equation (4.1) a deterministic function of the variational parameters, so as to be able to carry out gradient-based optimization despite the stochasticity of the objective. Considering input vectors $\mathbf{h}_i$ to a given layer, an improvement over this approach is to consider the distribution of the product $\mathbf{W}\mathbf{h}_i$. This is also known as the *local reparameterization trick* (Kingma and Ba, 2015), and it reduces the variance of stochastic gradients in the optimization, thus improving convergence. The product $\mathbf{W}\mathbf{h}_i$ follows the distribution $\mathcal{N}(\mathbf{m}, \mathbf{A}\mathbf{A}^\top)$ (A. K. Gupta and Nagar, 1999), with

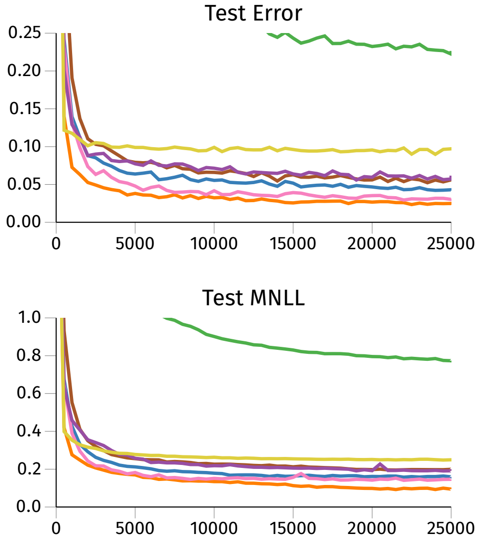
$$\mathbf{m} = \mathbf{S}_1\mathbf{H}\mathrm{diag}(\boldsymbol{\mu})\mathbf{H}\mathbf{S}_2\mathbf{h}_i, \quad \text{and} \quad \mathbf{A} = \mathbf{S}_1\mathbf{H}\mathrm{diag}(\mathbf{H}\mathbf{S}_2\mathbf{h}_i)\boldsymbol{\Sigma}^{1/2}. \tag{4.14}$$

A sample from this distribution can be efficiently computed thanks to the Walsh-Hadamard transform as: $\overline{\mathbf{W}}(\boldsymbol{\mu})\mathbf{h}_i + \overline{\mathbf{W}}(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\varepsilon})\mathbf{h}_i$, with $\overline{\mathbf{W}}$ a linear matrix-valued function $\overline{\mathbf{W}}(\mathbf{u}) = \mathbf{S}_1\mathbf{H}\mathrm{diag}(\mathbf{u})\mathbf{H}\mathbf{S}_2$.

4.4 ALTERNATIVE STRUCTURES, TENSOR FACTORIZATION AND EXTENSIONS

The choice of the parameterization of $\mathbf{W}$ in WHVI leaves space to several possible alternatives, which we compare in Figure 4.3. For all of them, $\mathbf{G}$ is learned variationally and the remaining diagonal $\mathbf{S}_i$ (if any) are either optimized or treated variationally (Gaussian mean-field). Figure 4.3 shows the behavior of these alternatives when applied to a 2×64 network with rectified linear unit (ReLU) activations. With the exception of the simple and highly constrained alternative $\mathbf{G}\mathbf{H}$, all parameterizations are converging quite easily and the comparison with MCD shows that indeed the proposed WHVI performs better both in terms of error rate and MNLL. Furthermore, this result allows us to speculate that our proposal is the best.

WHVI is effectively imposing a factorization of $\mathbf{W}$, where parameters are either optimized or treated variationally. Tensor decompositions for DNNs and CNNs have been proposed in (Novikov et al., 2015); here $\mathbf{W}$ is decomposed into k small matrices



Model	Test Error	Test MNLL
MCD	0.097	0.249
GH	0.226	0.773
$S_{\text{var}}HGH$	0.043	0.159
$S_{1,\text{var}}HGHS_{2,\text{var}}H$	0.061	0.190
$S_{\text{opt}}HGH$	0.054	0.199
$S_{1,\text{opt}}HGHS_{2,\text{opt}}H$	0.031	0.146
$S_{1,\text{opt}}HGHS_{2,\text{opt}}$ (WHVI)	**0.026**	**0.094**

Figure 4.3: Ablation study of different structures for the parameterization of the weights distribution. Metric: test error rate and test mean negative loglikelihood (MNLL) with different structures for the weights. Benchmark on Drive with a 2 × 64 network. Colors are coded to match the ones used in the Figure

(tensor cores), such that $W = W_1 W_2 \cdots W_k$, where each W_i has dimensions $r_{i-1} \times r_i$ (with $r_1 = r_k = D$). We adapt this idea to make a comparison with WHVI. In order to match the space and time complexity of WHVI, assuming $\{r_i = R | \forall i = 2, \dots, k-1\}$, we set: $R \propto \log_2 D$ and $K \propto \frac{D}{(\log_2 D)^2}$. Also, to match the number of variational parameters, all internal cores ($i = 2, \dots, k-1$) are learned with fully factorized Gaussian posterior, while the remaining are optimized (see Table 4.1). Given the same asymptotic complexity, Figure 4.4 reports the results of this comparison again on a 2 hidden layer network. Not only WHVI can reach better solutions in terms of test performance, but optimization is also faster. We speculate that this is attributed to the redundant variational parameterization induced by the tensor cores, which makes the optimization landscapes highly multi-modal, leading to slow convergence.

4.4.1 Extensions

CONCATENATING OR RESHAPING PARAMETERS. For the sake of presentation, so far we have assumed $W \in \mathbb{R}^{D \times D}$ with $D = 2^d$, but we can easily extend WHVI to handle parameters of any shape $W \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$. One possibility is to use WHVI with a large $D \times D$ matrix with $D = 2^d$, such that a subset of its elements represent W. Alternatively, a suitable value of d can be chosen so that W is a concatenation by

Table 4.1: Time and space complexity of various approaches to VI. Note: D is the dimensionality of the feature map, K is the number of tensor cores, R is the rank of tensor cores and M is the number of pseudo-data used to sample from a matrix Gaussian distribution, see Louizos and Welling (2016).

	Complexity	
	Space	Time
Mean field Gaussian	$\mathcal{O}(D^2)$	$\mathcal{O}(D^2)$
Gaussian matrix variate	$\mathcal{O}(D^2)$	$\mathcal{O}(D^2 + M^3)$
Tensor factorization	$\mathcal{O}(KR^2)$	$\mathcal{O}(R^2)$
WHVI	$\mathcal{O}(D)$	$\mathcal{O}(D \log D)$

Algorithm 3: Setup dimensions for non-squared matrix

```
Function SetupDimensions($D_{\text{in}}, D_{\text{out}}$):
    next power $\leftarrow 2^{\text{ceillog}_2 D_{\text{in}}}$;
    if next power == $2D_{\text{in}}$ then
        padding $\leftarrow 0$;
    else
        padding $=$ next power $- D_{\text{in}}$;
        $D_{\text{in}} \leftarrow$ next power;
    stack, remainder = divmod($D_{\text{out}}$, $D_{\text{in}}$);
    if remainder != 0 then
        stack $\leftarrow$ stack $+ 1$;
        $D_{\text{out}} \leftarrow D_{\text{in}} \times$ stack;
    return $D_{\text{in}}$, $D_{\text{out}}$, padding, stack
```

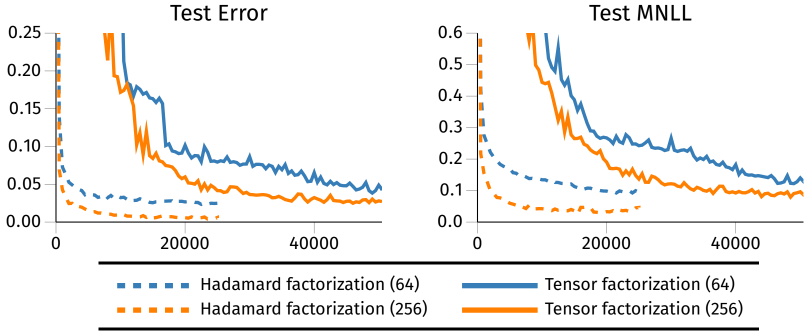


Figure 4.4: Comparison between Hadamard factorization in WHVI and tensor factorization. The number in the parenthesis is the hidden dimension. Plot is w.r.t. iterations rather then time to avoid implementation artifacts. The dataset used is Drive.

row/column of square matrices of size $D = 2^d$, padding if necessary (Algorithm 3 shows this case). When one of the dimensions is equal to one so that the parameter matrix is a vector ($W = w \in \mathbb{R}^D$), this latter approach is not ideal, as WHVI would fall back on mean-field VI. WHVI can be extended to handle these cases efficiently by reshaping the parameter vector into a matrix of size 2^d with suitable d, again by padding if necessary. Thanks to the reshaping, WHVI uses $\sqrt{D}$ parameters to model a posterior over D, and allows for computations in $\mathcal{O}(\sqrt{D} \log D)$ rather than D. This is possible by reshaping the vector that multiplies the weights in a similar way.

NORMALIZING FLOWS. Normalizing flows (D. Rezende and Mohamed, 2015) are a family of parameterized distributions that allow for flexible approximations. In the general setting, consider a set of invertible, continuous and differentiable functions $f_k : \mathbb{R}^D \to \mathbb{R}^D$ with parameters λ_k. Given $\theta_0 \sim q_0(\theta_0)$, θ_0 is transformed with a chain of K flows to $\theta_K = (f_K \circ \cdots \circ f_1)(\theta_0)$. The variational lower bound slightly differs from Equation (4.1) to take into account the determinant of the Jacobian of the transformation, yielding a new variational objective as follows:

$$\mathbb{E}_q \log p(y \mid \theta, X) - \mathrm{KL}\left[q_0(\theta_0) \parallel p(\theta_K)\right] + \mathbb{E}_{q_0(\theta_0)}\left[\sum_{k=1}^{K} \log \left|\det \frac{\partial f_k(\theta_{k-1}; \lambda_k)}{\partial \theta_{k-1}}\right|\right]. \tag{4.15}$$

Setting the initial distribution q_0 to a fully factorized Gaussian $\mathcal{N}(\theta_0 \mid \mu, \sigma \mathbf{I})$ and assuming a Gaussian prior on the generated θ_K, the KL term is still tractable. The tranformation f is generally chosen to allow for fast computation of the determinant of the Jacobian. The parameters of the initial density q_0 as well as the flow parameters λ are optimized. In our case, we consider q_K as a distribution over the elements of $\mathbf{g}$. This approach increases the flexibility of the form of the variational posterior in WHVI, which is no longer Gaussian, while still capturing covariances across weights. This is obtained at the expense of losing the possibility of employing the local reparameterization trick. In the remaining part of the chapter, we will use *planar flows* (D. Rezende and Mohamed, 2015). Although this is a simple flow parameterization, a planar flow requires only $\mathcal{O}(D)$ parameters and thus it does not increase the time/space complexity of WHVI. More complex alternatives are proposed by (Van den Berg et al., 2018; Kingma, Salimans, et al., 2016; Louizos and Welling, 2017).

4.5 EMPIRICAL EVALUATION

In this section we will provide experimental evaluations of our proposal, with experiments ranging from regression on classic benchmark datasets to image classification with large-scale convolutional neural networks. We will also comment on the computational efficiency and some potential limitation of our proposal.

SETUP. The experiments on Bayesian DNN are run with the following setup. For WHVI, we used a zero-mean prior over $\mathbf{g}$ with fully factorized covariance $\lambda\mathbf{I}$; $\lambda = 10^{-5}$ was chosen to obtain sensible variances in the output layer. It is possible to design a prior over $\mathbf{g}$ such that the prior on $\mathbf{W}$ has constant marginal variance and low correlations although empirical evaluations showed not to yield a significant improvement compared to the previous (simpler) choice. In the final implementation of WHVI that we used in all experiments, $\mathbf{S}_1$ and $\mathbf{S}_2$ are optimized. We used classic Gaussian likelihood with optimized noise variance for regression and softmax likelihood for classification. Training is performed for 500 steps with fixed noise variance and for other 50000 steps with optimized noise variance. Batch size is fixed to 64 and for estimation of the stochastic gradients we use one sample at train time and 64 samples at test time. We choose the Adam optimizer (Kingma and Ba, 2015) with exponential learning rate decay $\lambda_{t+1} = \lambda_0(1+\gamma t)^{-p}$, with $\lambda_0 = 0.001$, $p = 0.3$, $\gamma = 0.0005$ and t being the current iteration. Similar setup was also used for the Bayesian CNN experiment. The only differences are the batch size – increased to 256 – and the optimizer, which is run without learning rate decay.

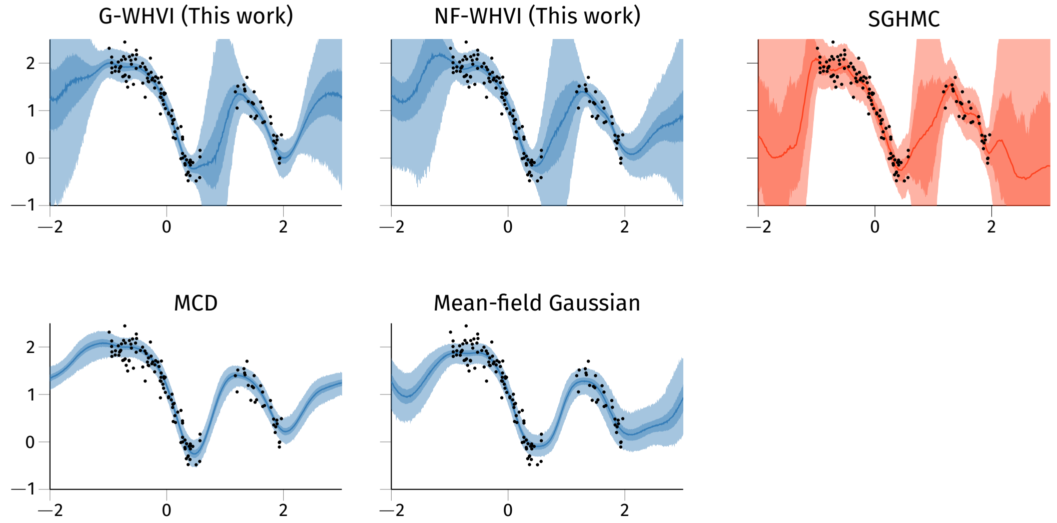


Figure 4.5: Regression example trained using WHVI with Gaussian vector (1541 param.) and with planar normalizing flow (10 flows for a total of 4141 param.), mean-field Gussian (35k param.) and Monte Carlo dropout (MCD) (17k param.). The two shaded areas represent the 95th and the 75th percentile of the predictions. As "ground truth", we also show the predictive posterior obtained by running Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) on the same model ($R < 1.05$, (Gelman et al., 2004)).

4.5.1 Toy example

We begin our experimental validation with a 1D-regression problem. We generated a 1D toy regression problem with 128 inputs sampled from $\mathcal{U}[-1, 2]$, and removed 20% inputs on a predefined interval; targets are noisy realizations of a random function (noise variance $\sigma^2 = \exp(-3)$). We model these data using a DNN with 2 hidden layers of 128 features and cosine activations. We test four models: variational inference with mean-field Gaussian, Monte Carlo dropout (MCD, Gal and Ghahramani, 2016b) with dropout rate 0.4 and two variants of WHVIs – G-WHVI with Gaussian posterior and NF-WHVI with planar flows (10 planar flows). We also show the free form posterior obtained by running a Markov chain Monte Carlo (MCMC) algorithm, SGHMC in this case (T. Chen et al., 2014; Springenberg et al., 2016), for several thousands steps. As Figure 4.5 shows, WHVIs offers a sensible modeling of the uncertainty on the input domain, whereas mean-field and MCD seem to be slightly over-confident.

4.5.2 Empirical comparison on the UCI benchmark

We conduct now a series of comparisons with state-of-the-art VI schemes for Bayesian DNNs; see the Supplement for the list of data sets used in the experiments.

Table 4.2: Test root mean squared error (RMSE) and test MNLL for regression datasets. Results in the format "*mean (std)*"

	Test error				Test MNLL			
Model Dataset	MCD	MFG	NNG	WHVI	MCD	MFG	NNG	WHVI
Boston	3.91 (0.86)	4.47 (0.85)	3.56 (0.43)	**3.14** (0.71)	6.90 (2.93)	2.99 (0.41)	**2.72** (0.09)	4.33 (1.80)
Concrete	5.12 (0.79)	8.01 (0.41)	8.21 (0.55)	**4.70** (0.72)	3.20 (0.36)	3.41 (0.05)	3.56 (0.08)	**3.17** (0.37)
Energy	2.07 (0.11)	3.10 (0.14)	1.96 (0.28)	**0.58** (0.07)	4.15 (0.15)	4.91 (0.09)	2.11 (0.12)	**2.00** (0.60)
Kin8nm	0.09 (0.00)	0.12 (0.00)	**0.07** (0.00)	0.08 (0.00)	−0.87 (0.02)	−0.83 (0.02)	**-1.19** (0.04)	**-1.19** (0.04)
Naval	0.30 (0.30)	0.01 (0.00)	**0.00** (0.00)	0.01 (0.00)	−1.00 (2.27)	−6.23 (0.01)	**-6.52** (0.09)	−6.25 (0.01)
Powerplant	**3.97** (0.14)	4.52 (0.13)	4.23 (0.09)	4.00 (0.12)	2.74 (0.05)	2.83 (0.03)	2.86 (0.02)	**2.71** (0.03)
Protein	**4.23** (0.10)	4.93 (0.11)	4.57 (0.47)	4.36 (0.11)	**2.76** (0.02)	2.92 (0.01)	2.95 (0.12)	2.79 (0.01)
Yacht	1.90 (0.54)	7.01 (1.22)	5.16 (1.48)	**0.69** (0.16)	2.95 (1.27)	3.38 (0.29)	3.06 (0.27)	**1.80** (1.01)

We compare WHVI with MCD and NNG (NOISY-KFAC, G. Zhang et al., 2018). MCD draws on a formal connection between dropout and VI with Bernoulli-like posteriors, while the more recent NOISY-KFAC yields a matrix-variate Gaussian distribution using noisy natural gradients. To these baselines, we also add the comparison with mean field Gaussian (MFG). In WHVI, the last layer assumes a fully factorized Gaussian posterior.

Data is randomly divided into 90%/10% splits for training and testing eight times. We standardize the input features x while keeping the targets y unnormalized. Differently from the experimental setup in Louizos and Welling (2016), G. Zhang et al. (2018), and Hernandez-Lobato and R. Adams (2015), we use the same architecture regardless of the size of the dataset. Futhermore, to test the efficiency of WHVI in case of over-parameterized models, we set the network to have two hidden layers and 128 features with ReLU activations (as a reference, these models are ~20 times bigger than the usual setup, which uses a single hidden layer with 50/100 units). We report the test RMSE and the average predictive test negative log-likelihood (MNLL) in Table 4.2. On the majority of the datasets, WHVI outperforms MCD and NOISY-KFAC, while requiring less parameters to be optimized. Futhermore, we study how the test MNLL varies with the number of hidden units in a 2-layered network. As Figure 4.6 shows, WHVI and MCD both behave well while the other competitive methods struggle. Empirically, these results demonstrate the value of WHVI, which offers a competitive parameterization of a matrix-variate Gaussian posterior.

Being able to increase width and depth of a model without drastically increasing the number of variational parameters is also one of the competitive advantages of

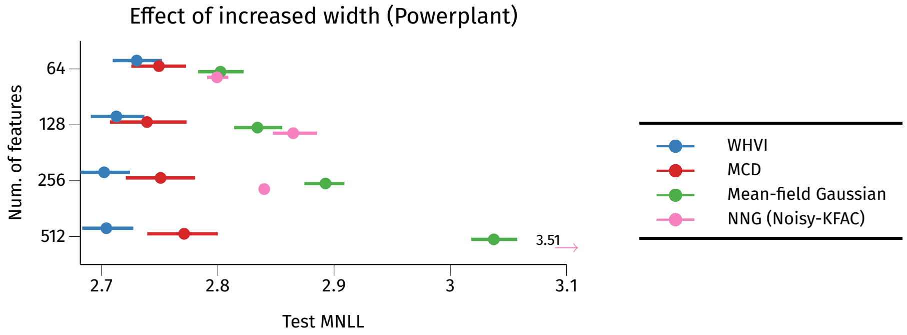


Figure 4.6: Comparison of the test MNLL as a function of the number of hidden units. The dataset used is Powerplant.

WHVI. Using classification datasets, we now study the effect of changing depth and width of the architecture. Figure 4.7 reports a summary of this comparison, where we considere two and three hidden layer networks with width from 64 up to 512. While reasoning about the role of overparameterization in deep and wide neural networks for increasing generalization is beyond the scope of this work, we observe that at test time increasing the number of hidden layers and the numbers of hidden features WHVI allows the model to deliver better performance, while avoiding catastrofic overfitting. This is not always the case for e.g. MCD, which with few exceptions has performance degradation the wider and deeper a model is.

We report the training and test curves for Mocap in Figure 4.8. While the analysis of the optimization problem is not a primary focus of this work, the behavior of the training evidence lower bound (ELBO) reported shows stable convergence. Thanks to the structure of the weights matrices, widening and deepening the model is beneficial performance-wise and doesn't incur in convergence instability observed for other methods. Furthermore, with a non-mean field assumption of the weights, the ELBO is tighter than before and, in principle, it could be used as a suitable objective function for model selection.

4.5.3 Bayesian convolutional neural networks for image classification

We continue the experimental evaluation of WHVI by analyzing its performance on CNNs. For this experiment, we replace all fully-connected layers in the CNN with the WHVI parameterization, while the convolutional filters are treated variationally

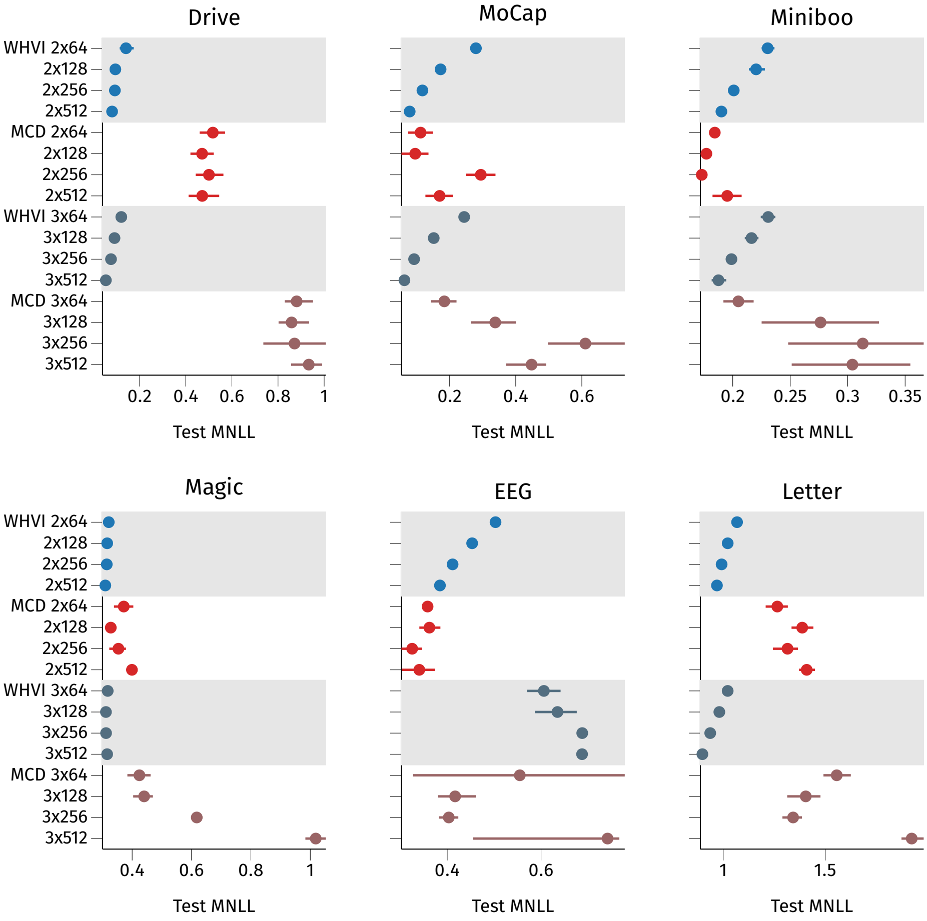


Figure 4.7: Comparison of the test MNLL as a function of the number of hidden units and hidden layers.

using MCD. In this setup, we fit VGG (Simonyan and Zisserman, 2014), ALEXNET (Krizhevsky, Sutskever, et al., 2012) and RESNET -18 (K. He et al., 2016) on CIFAR10 . Using WHVI, we can reduce the number of parameters in the linear layers without affecting neither test performance nor calibration properties of the resulting model, as shown in Figure 4.9 and Table 4.3. For ALEXNET and RESNET we also try our variant of WHVI with normalizing flows. Even though we lose the benefits of the local reparameterization, the higher flexibility of normalizing flows allows the model to obtain better test performance with respect to the Gaussian posterior. This can be improved even further using more complex families of normalizing flows (D. Rezende and Mohamed, 2015; Van den Berg et al., 2018; Kingma, Salimans, et al., 2016; Louizos and Welling, 2017). With WHVI, ALEXNET and its original ~23.3M parameters is reduced to just ~2.3M (9.9%) when using G-WHVI and to ~2.4M (10.2%) with WHVI and 3 planar flows.

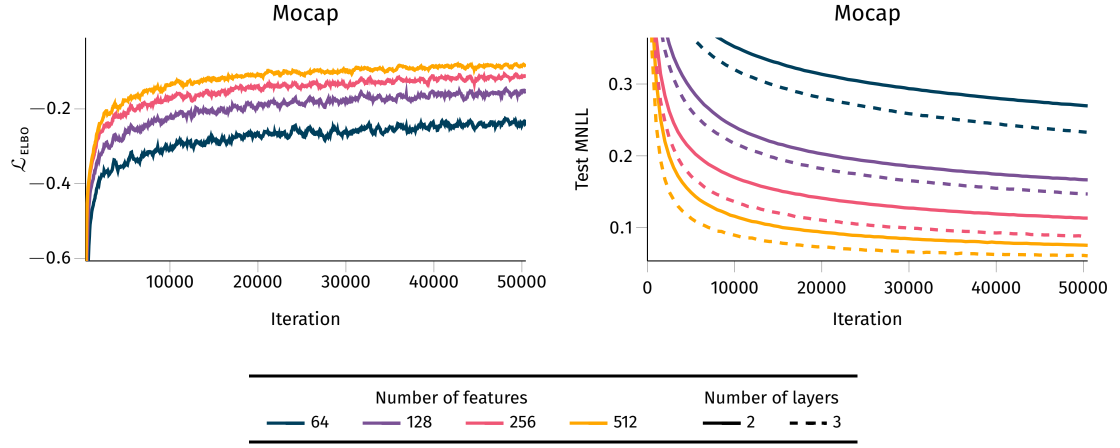


Figure 4.8: Analysis of the training curve (on the left) and test curve (on the right) for the Mocap dataset.

WHVI FOR CONVOLUTIONAL LAYERS. Convolution layers are linear layers applied on a series of patches extracted from the image and they can be implemented with matrix multiplication, once filters are reshaped in 2D. Given this interpretation of the convolutional layers, we also extended WHVI for for this case. We observe though that in this case resulting models had too few parameters to obtain any interesting results. For ALEXNET , we obtained a model with just 189K parameters, which corresponds to a sparsity of 99.2% with respect of the original model. As a reference, Wen et al. (2016) was able to reach sparsity only up to 60% in the convolutional layers without impacting performance. To study this behavior in details, we take a simple CNN with two convolutional layers and one linear layer (Figure 4.10). We see that the combination of MCD and WHVI performs very well in terms of convergence and test performance, while the use of WHVI on the convolutional filters brings an overall degradation of the performance. Interestingly, though, we also observe that MCD with the same number of parameters as for WHVI (referred to as low-rank MCD) performs even worse than the baseline: this once again confirms the parameterization of WHVI as an efficient alternative.

4.5.4 Comments on computational efficiency

WHVI builds his computational efficiency on the Fast Walsh-Hadamard Transform (FWHT), which allows to cut the complexity of a D-dimensional matrix-vector multiplication $\mathbf{H}\mathbf{x}$ from a naive $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$, without generating and storing $\mathbf{H}$. For our experimental evaluation, we implemented a batched version of the FWHT in PyTorch in C++ and CUDA to leverage the full computational capabilities of

Table 4.3: Test performance of different Bayesian CNNs.

	CIFAR10	Test error	Test MNLL
VGG16	MFG	16.82%	0.6443
	MCD	21.47%	0.8213
	Noisy-KFAC	15.21%	**0.6374**
	WHVI	**12.85%**	0.6995
AlexNet	MFG	–	–
	MCD	13.30%	0.9590
	Noisy-KFAC	20.36%	1.1990
	WHVI	13.56%	**0.6164**
	NF-WHVI	**12.72%**	0.6596
ResNet18	MFG	–	–
	MCD	**10.71%**	0.8468
	Noisy-KFAC	–	–
	WHVI	11.46%	0.5513
	NF-WHVI	11.42%	**0.4908**

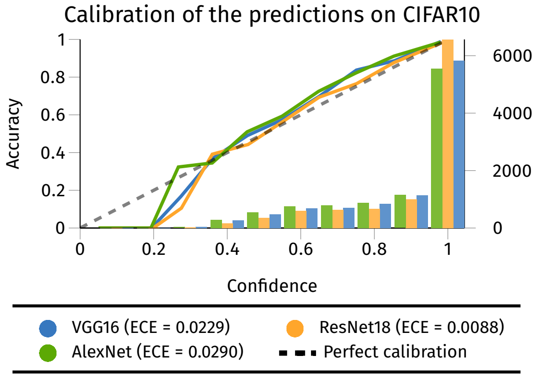


Figure 4.9: Reliability diagram and expected calibration error of VGG , ALEXNET and RESNET with WHVI.

modern graphic processing units (GPUs). Figure 4.11a presents a timing profiling of our implementation versus the naive `matmul` (batch size of 512 samples and profiling repeated 1000 times). The breakeven point for the CPU implementation is in the neighborhood of 512/1024 features, while on GPU we see that FWHT is consistently faster. In Figure 4.11b we also compare the time required to sample and compute the predictions on the test set on two regression datasets as a function of the number of hidden units in a two-layer DNN. The workstation used is equipped with two Intel Xeon CPUs, four NVIDIA Tesla P100 and 512 GB of RAM. Each experiment is carried out on a GPU fully dedicated to it. The NNG algorithm is implemented in Tensorflow [1] while the others are written in PyTorch. We made sure to fully exploit all parallelization opportunities in the competiting methods and ours; we believe that the timings are not severely affected by external factors other than the actual implementation of the algorithms. We also report the storage footprint (in Bytes) for several configuration of depth/width in Figure 4.12. To put things into prospective, the test MNLL is also reported. The Pareto analysis shows that for most of the cases WHVI shows good trade-off between number of parameters and modeling performance. Indeed, there are configurations for which MCD delivers better test MNLL, but all of them are at the expeses of bigger models.

Finally, in Figure 4.13, we analyze the energy consumption required to sample from the converged model and predict on the test set of CIFAR10 with ALEXNET

---

[1] `github.com/gd-zhang/noisy-K-FAC` — `github.com/pomonam/NoisyNaturalGradient`

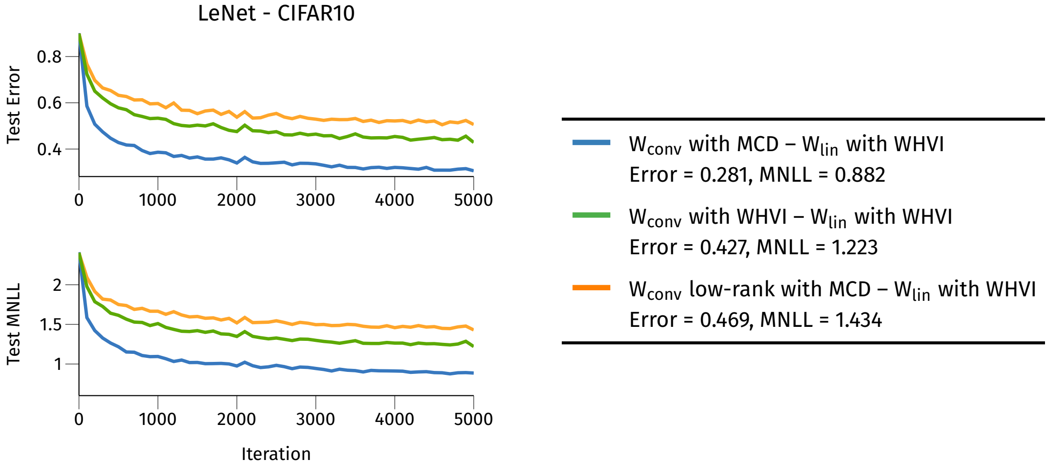


Figure 4.10: Test curve of LeNet when trained with different combination of variational inference parameterization.

using WHVI and MCD and noisy natural gradients. The regularity of the algorithm for computing the FWHT and its reduced memory footprint result on an overall higher utilization of the GPU, 85% for WHVI versus ~ 70% for MCD. This translates into an increase of energy efficiency up to 33% w.r.t MCD, despite being 51% faster. We speculate that the poor performance of NNG is due to the inversion of the approximation to the Fisher matrix, which scales cubically in the number of units.

4.5.5 Exploring the parameter efficiency of WHVI with Gaussian processes

We conclude this empirical evaluation with a discussion on a possible application of WHVI for scalable Gaussian processs (GPs). GPs represent the perfect controlled environment where we can analyze the trade off between model approximation and inference approximation, with a focus on parameter efficiency. To this extent, we focus on GPs with random feature expansions (Lázaro-Gredilla et al., 2010). Given a random feature expansion of the kernel martix, say $\mathbf{K} \approx \mathbf{\Phi}\mathbf{\Phi}^\top$, the latent variables can be rewritten as $\mathbf{f} = \mathbf{\Phi}\boldsymbol{w}$, with $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The random features $\mathbf{\Phi}$ are constructed by randomly projecting the input matrix $\mathbf{X}$ using a Gaussian random matrix $\mathbf{\Omega}$ and applying a nonlinear transformation, which depends on the choice of the kernel function. For a given set of random features, we can treat the weights of the resulting model variationally, with e.g. a mean field Gaussian $q(\boldsymbol{w})$. By reshaping the vector of parameters $\boldsymbol{w}$ into a $D \times D$ matrix, WHVI allows for the

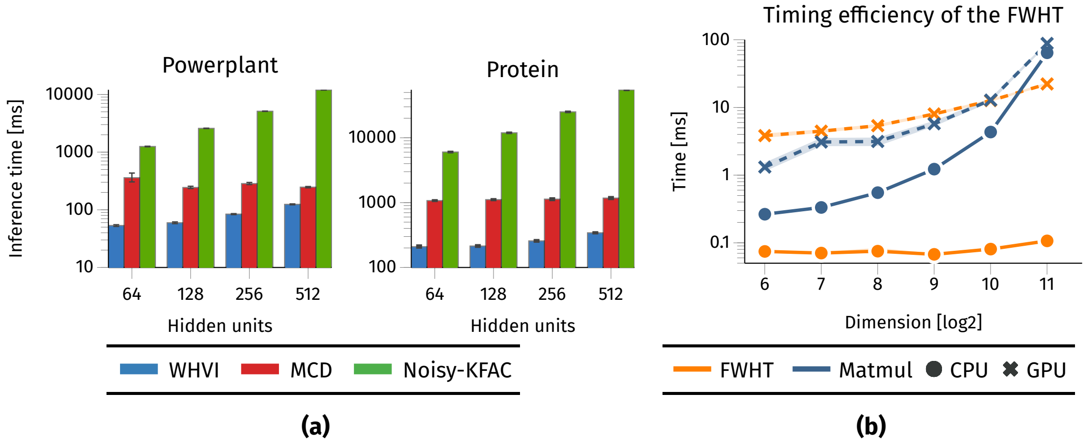


Figure 4.11: Analysis of the timing efficiency of WHVI. **(a)** Inference time on the test set with 128 batch size and 64 Monte Carlo samples. Experiment repeated 100 times on an GPU fully dedicated to it. **(b)** Timing comparison of different implementations of the vectorized Fast Walsh-Hadamard transform, with a batch size of 512 data points.

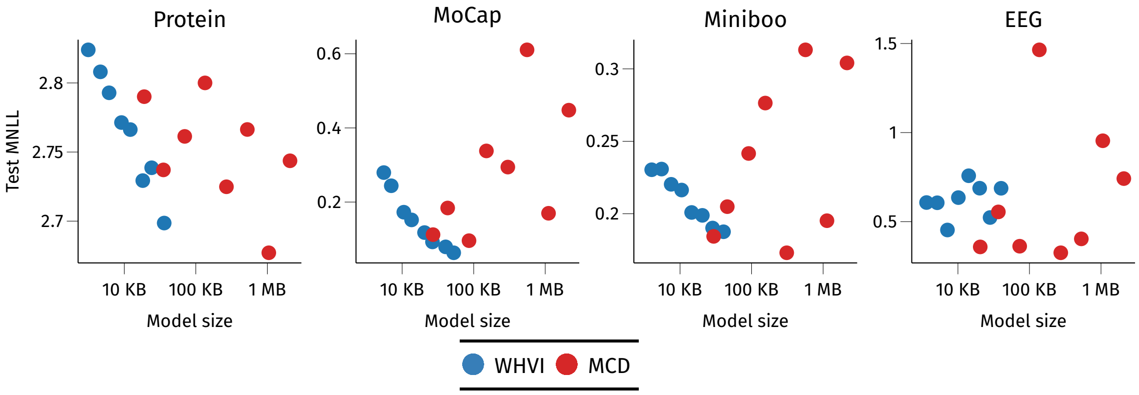


Figure 4.12: Comparison of the test MNLL as a function of the memory footprint for four datasets.

linearized GP to reduce the number of parameters to optimize or, given the same complexity, increase the number of random features (see **??**). This latter case will be also compared with another approximation of Gaussian process using variational inference and inducing points (Titsias, 2009b; Hensman, A. G. Matthews, et al., 2015). We report the results on four datasets in Figure 4.14. This variant of WHVI that reshapes w into a matrix largely outperforms the direct variational inference with mean field Gaussian. However, it appears that this improvement of the random feature inference for GPs is not enough to reach the performance of inducing points. Inducing point approximations are based on the Nyström approximation of kernel matrices, which are known to lead to lower approximation error on the elements on the kernel matrix compared to random features approximations. This is the

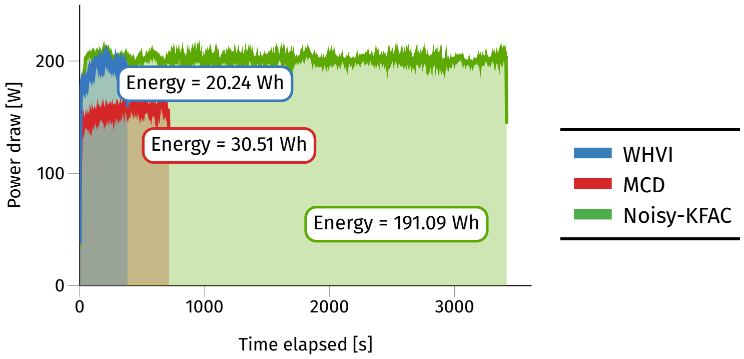


Figure 4.13: Power profiling during inference on the test set of CIFAR10 with ALEXNET The task is repeated 16 consecutive times and profiling is carried out using the `nvidia-smi` tool.

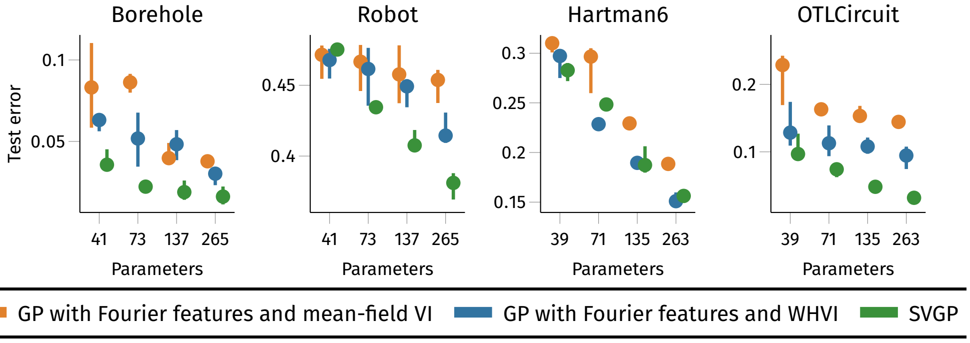


Figure 4.14: Analysis of the parameter efficiency of WHVI for scalable Gaussian process regression with variational inference.

reason we attribute to the lower performance of WHVI compared to inducing points approximations in this experiment.

4.6 RELATED WORK

In the early sections of the chapter, we have already briefly reviewed some of the literature on VI and Bayesian DNNs and CNNs; here we complement the literature by including other relevant works that have connections with WHVI.

Our work takes inspiration from the works on random features for kernel approximation (Rahimi and Recht, 2008) and Fastfood (Le et al., 2013). Random feature expansions have had a wide impact on the literature on kernel methods. Such approximations have been successfully used to scale a variety of models, such as Support Vector Machines (Rahimi and Recht, 2008), Gaussian processes

(Lázaro-Gredilla et al., 2010) and Deep Gaussian processes (Cutajar et al., 2017; Gal and Ghahramani, 2016b). This has contributed to bridging the gap between Deep GPs and Bayesian DNNs and CNNs (Neal, 1996; Duvenaud et al., 2014; Cutajar et al., 2017; Gal and Ghahramani, 2016a), which is an active area of research which aims to gain a better understanding of deep learning models through the use of kernel methods (G. Matthews et al., 2018; Dunlop et al., 2018; Garriga-Alonso et al., 2019). Structured random features (Le et al., 2013; F. X. Yu et al., 2016; Bojarski et al., 2017) have been also applied to the problem of handling large dimensional convolutional features (Z. Yang et al., 2015) and Convolutional GPs (G.-L. Tran et al., 2019).

Bayesian inference on DNNs and CNNs has been research topic of several seminar works (see e.g. Graves, 2011; Hernandez-Lobato and R. Adams, 2015; Blundell et al., 2015; Gal and Ghahramani, 2016b; Gal and Ghahramani, 2016a). Recent advances in DNNs have investigated the effect of over-parameterization and how model compression can be used during or after training (Hubara et al., 2016; Louizos, Ullrich, et al., 2017; Zhu and S. Gupta, 2018). Our current understanding shows that model performance is affected by the network size with bigger and wider neural networks being more resilient to overfit (Neyshabur, Tomioka, et al., 2015; Neyshabur, Z. Li, et al., 2019). For variational inference, and Bayesian inference in general, over-parameterization is reflected on over-regularization of the objective, leading the optimization to converge to trivial solutions (posterior equal to prior). Several works have encountered and proposed solutions to such issue (Higgins et al., 2017; Burgess et al., 2018; Bowman et al., 2016; Sønderby et al., 2016; Rossi, Michiardi, et al., 2019). The problem of how to run accurate Bayesian inference on over-parametrized models like Bayesian neural networks (BNNs) is still an ongoing open question (A. G. Wilson and Izmailov, 2020a; Wenzel et al., 2020)

4.7 FINAL REMARKS

Inspired by the literature on scalable kernel methods, this chapter proposed Walsh-Hadamard Variational Inference (WHVI). WHVI offers a novel parameterization of the variational posterior, which is particularly attractive for over-parameterized models, such as modern DNNs and CNNs. WHVI assumes a matrix-variate posterior distribution, which therefore captures covariances across weights. Crucially, unlike previous work on matrix-variate posteriors for VI, this is achieved with a light parameterization and fast computations, bypassing the over-regularization issues of VI for over-parameterized models. The large experimental campaign, demonstrates that WHVI is a strong competitor of other variational approaches for such models, while offering considerable speedups. One limitation of the parameterization induced by WHVI is that its mean cannot span the whole D^2-dimensional space. A simple and cheap remedy could be to modify the parameterization from $\mathbf{S}_1\mathbf{H}\mathbf{G}\mathbf{H}\mathbf{S}_2$ to $\mathbf{S}_1\mathbf{H}\mathbf{G}\mathbf{H}\mathbf{S}_2 + \mathbf{M}$ with $\mathbb{E}[\mathbf{G}] = \mathbf{0}$ so that the mean can span the whole space thanks to $\mathbf{M}$, while the rest would allow to model covariances across weights. However, though a preliminary numerical study on the RMSE between the weights induced by WHVI and arbitrary weight matrices, we see constant behavior w.r.t. D. Possible extensions include the possibility to capture the covariance between weights across layers, by either sharing the matrix $\mathbf{G}$ across, or by concatenating all weights into a single matrix which is then treated using WHVI, with the necessary adaptations to handle the sequential nature of computations.

5

THE EFFECT OF SELECTING THE PRIOR FOR BAYESIAN DEEP LEARNING

The Bayesian treatment of neural networks dictates that a prior distribution is specified over their weight and bias parameters. This poses a challenge because modern neural networks are characterized by a large number of parameters, and the choice of these priors has an uncontrolled effect on the induced functional prior, which is the distribution of the functions obtained by sampling the parameters from their prior distribution. We argue that this is a hugely limiting aspect of Bayesian deep learning, and this chapter tackles this limitation in a practical and effective way.

5.1 THE CHOICE OF THE PRIOR MATTERS

The concept of prior distribution in Bayesian inference allows us to describe the family of solutions that we consider acceptable, *before* having seen any data. While in some cases selecting an appropriate prior is easy or intuitive given the context (O'Hagan, 1991; Rasmussen and Ghahramani, 2002; Srinivas et al., 2010; Cockayne et al., 2019; Briol et al., 2019), for nonlinear parametric models with thousands (or millions) of parameters, like deep neural networks (DNNs) and convolutional neural networks (CNNs), this choice is not straightforward. Despite many advances in the field (Kendall and Gal, 2017; Rossi, Michiardi, et al., 2019; Osawa et al., 2019; Rossi, Marmin, et al., 2020), it is reported that in some cases the predictive posteriors are not competitive to non-Bayesian alternatives, making these

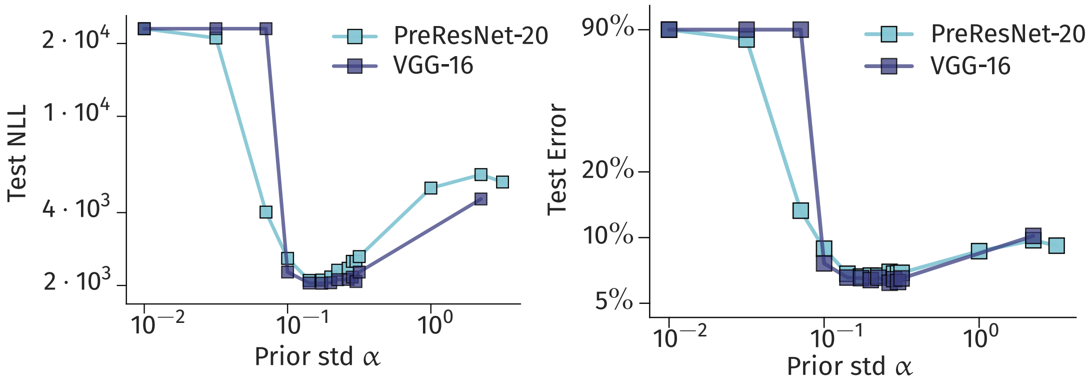


Figure 5.1: Test performance on CIFAR10 with different priors. Figure replicated from the arXiv version of A. G. Wilson and Izmailov (2020a).

models—and Bayesian deep learning, in general—less than ideal solutions for a number of applications. Wenzel et al. (2020), for example, raise concerns about the quality and the usefulness of Bayesian neural network (BNN) posteriors, where it is found that tempering the posterior distribution improves the performance of some deep models. Observations of this kind should not be really surprising. Bayesian inference is a recipe with exactly three ingredients: the prior distribution, the likelihood and the Bayes' rule. Regarding the Bayes' rule, that is simply a consequence of the axioms of probability. The fact that the posterior might not be useful in some cases should never be attributed to the Bayesian method itself. In fact, it is very easy to construct Bayesian models with poor priors and/or likelihoods, which result in poor predictive posteriors. One should therefore turn to the other two components, which encode model assumptions.

In this chapter we will discuss the role of the priors in the predictive performance of BNNs. The common practice in the literature is to define a prior distribution on the network weights and biases, which is often chosen to be Gaussian. We argue that this choice can be catastrophic if miss-placed; nonetheless even with a simple Gaussian there are optimal configurations. In Figure 5.1, for example, A. G. Wilson and Izmailov (2020a) show the effect of the prior variance for two deep CNNs. Note that all weights and biases in all the layers share the same parameter, which—as we will see in a moment—might not be the best solution. From this analysis we can draw two conclusions: (i) priors have huge effect in the modeling task, (ii) there are configurations which are much better than the simple $\mathcal{N}(0, 1)$ priors but for which we don't grasp the intuition.

While these argument seams only logical, in practice they are not. One might expect, for instance, that provided with a sufficiently large amount of data, the

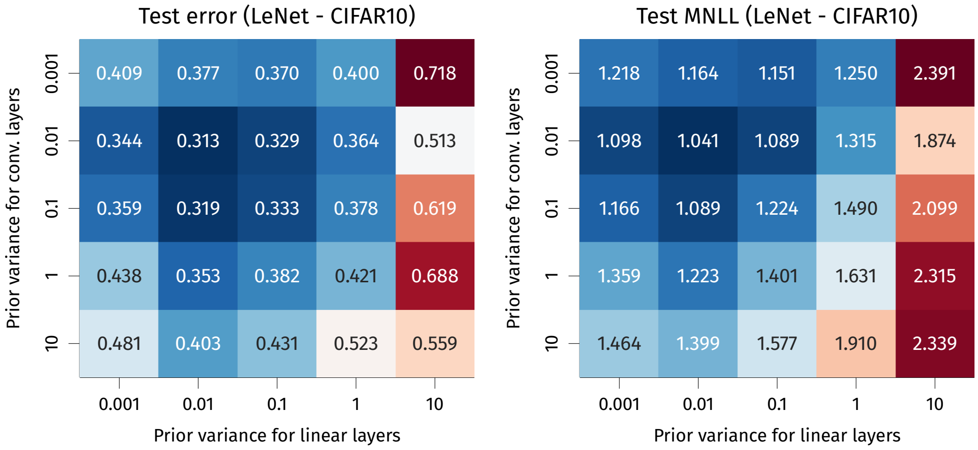


Figure 5.2: Grid search analysis on the prior variance for LeNet trained via variational inference on CIFAR10.

effect of the prior distribution will decrease. While this is true, the problem with Bayesian deep learning stands with the huge parameters dimensionality of the models considered, which pushes this limit further away. We show a simple case to exemplify this behaviour. In Figure 5.2 we consider a very simple LeNet classifier which is trained with variational inference and mean-field Gaussian posteriors. For different combination of prior variance for linear and convolutional layers we report the "converged" test error and test likelihood. As can be seen, it appears that the effect of the prior is more critical for linear layers rather then convolutional ones, which reflect the allocation of trainable parameters. This justify a careful analysis of the effect of the priors in deep convolutional and feedforward neural networks.

5.1.1 Pathologies of deep prior functions

A prior over the parameters induces a prior on the functions generated by the model, which also depends on the network architecture. However, due to the nonlinear nature of the model, the effect of this prior on the functional output is not obvious to characterize and control. Consider the example in Figure 5.3, where we show the functions generated by sampling the weights of BNNs with tanh activation from their Gaussian prior $\mathcal{N}(0, 1)$. We see that as depth is increased, the samples tend to form straight horizontal lines, which is a well-known pathology stemming from increasing model's depth (Neal, 1996; Duvenaud et al., 2014; G. Matthews et al., 2018). We stress that a fixed Gaussian prior on the parameters is not always problematic, but it can be, especially for deeper architectures. More recently, issues of these kind of BNN priors have been recently exposed by Wenzel et al., 2020, who suggest to consider a temperate version of the posterior—effectively reducing the strength of the prior—to increase performance. Other recent works (T. Chen et al., 2014; Springenberg et al., 2016) consider a hierarchical structure for the prior, where the variance of the normally-distributed BNN weights is governed by a Gamma distribution. This setting introduces additional flexibility on the space of functions, but it still does not provide much intuition regarding the properties of the prior.

Bayesian model selection constitutes a principled approach to select an appropriate prior distribution. Model selection is based on the marginal likelihood – the normalizing constant of the posterior distribution – which may be estimated from the training data. This practice is usually used to select hyperparameters of a Gaussian process (GP) as its marginal likelihood is available in closed form (Rasmussen and Williams, 2005). However, the marginal likelihood of BNNs is generally intractable, and lower bounds are difficult to obtain. Graves (2011) first and Blundell et al., 2015 later used the variational lower bound of the marginal likelihood for optimizing the parameters of a prior, yielding in some cases worse results. The Laplace's method is another alternative to approximate the marginal likelihood (Immer, Bauer, et al., 2021; D. J. MacKay, 1995) which is scalable and differentiable with respect to the prior hyperparameters.

Nonetheless, this kind of generative priors on the functions is very different from shallow Bayesian models, such as GPs, where the selection of an appropriate prior typically reflects certain attributes that we expect from the generated functions. A GP defines a distribution of functions that is characterized by a mean and a kernel function κ. The GP prior specification can be more *interpretable* than the one induced by the prior over the weights of a BNN, in the sense that the kernel effectively governs the properties of possible functions, such as shape, variability and smoothness. For example, shift-invariant kernels may impose a certain characteristic length-scale on the prior distribution over functions.

5.1.2 Contributions

The main research question that we investigate now is how to impose functional priors on BNNs. We seek to tune the prior distributions over BNNs parameters so that the induced functional priors exhibit interpretable properties, similar to shallow GPs. While BNN priors induce a regularization effect that penalizes large

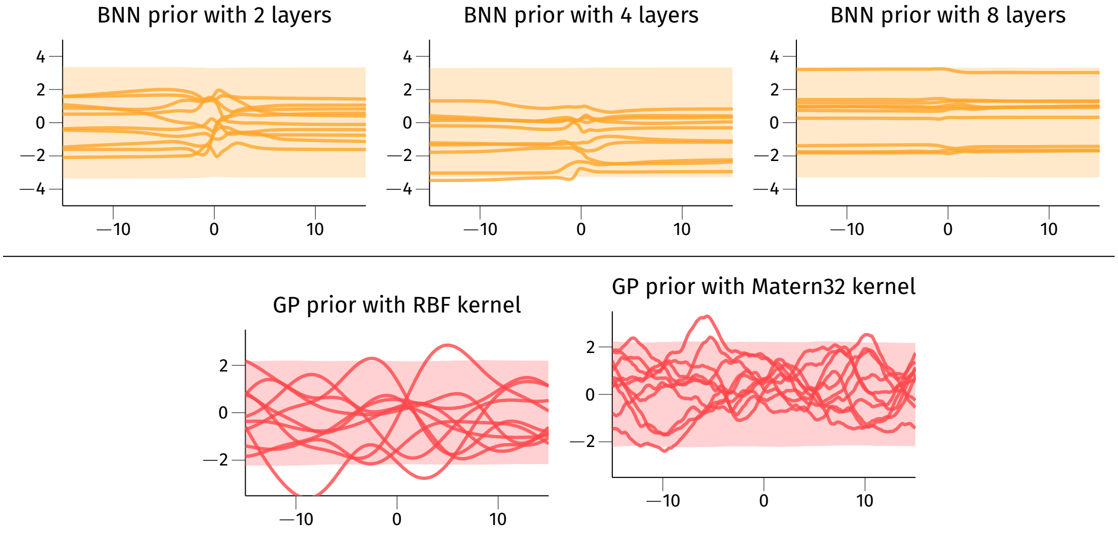


Figure 5.3: (**Top**) Sample functions of a fully-connected BNN with 2, 4 and 8 layers obtained by placing a Gaussian prior on the weights. (**Bottom**) Samples from a GP prior with two different kernels.

values for the network weights, a GP-adjusted prior induces regularization directly on the space of functions.

We consider the *Wasserstein distance* between the distribution of BNN functions induced by a prior over their parameters, and a target GP prior. We propose an algorithm that optimizes such a distance with respect to the BNN prior parameters and hyper-parameters. An attractive property of our proposal is that estimating the Wasserstein distance relies exclusively on samples from both distributions, which are easy to generate. We explore the effect of GP-induced priors on the predictive posterior distribution of BNNs by carrying out fully Bayesian inference of neural network models with these priors through the use of scalable Markov chain Monte Carlo (MCMC) sampling (T. Chen et al., 2014). We demonstrate systematic performance improvements over alternative choices of priors and state-of-the-art approximate Bayesian deep learning approaches, including convolutional neural networks.

5.2 THE PROBLEM OF CHOOSING PRIORS IN THE LITERATURE OF BAYESIAN DEEP LEARNING

Many recent attempts in the literature have turned their attention towards defining priors in the space of functions, rather than the space of weights. For example, Nalisnick et al. (2021) consider a family of priors that penalize the complexity of predictive functions. Hafner et al. (2019) propose a prior that is imposed on training inputs, as well as out-of-distribution inputs. This is achieved by creating pseudo-data by means of perturbing the training inputs; the posterior is approximated by a variational scheme. W. Yang et al. (2019) present a methodology to induce prior knowledge by specifying certain constraints on the network output. Pearce et al. (2019) explore DNN architectures that recreate the effect of certain kernel combinations for GPs. This result in an expressive family of network priors that converge to GPs in the infinite-width limit. A different approach is proposed by Karaletsos and T. Bui (2019) and Karaletsos and T. Bui (2020), who consider a GP model for the network parameters that can capture weight correlations.

A similar direction of research focuses not only on priors but also inference in the space of functions for BNNs. For example, Ma et al. (2019) consider a BNN as an implicit prior in function space and then use GP for inference. Conversely, Sun, G. Zhang, et al. (2019) propose a functional variational inference which employs a GP prior to regularize BNNs directly in the function space by estimating the Kullback-Leibler (KL) divergence between these two stochastic processes. However, this method relies on a gradient estimator which can be inaccurate in high dimensions. In another route, M. E. E. Khan et al. (2019) derive a GP posterior approximation for neural networks by means of the Laplace and generalized Gauss-Newton (GGN) approximations, leading to an implicit linearization. Immer, Korzepa, et al. (2021) make this linearization explicit and apply it to improve the performance of BNN predictions. In general, these approaches either rely heavily on non-standard inference methods or are constrained to use a certain approximate inference algorithm such as variational inference or Laplace approximation.

A different line of work focuses on meta-learning by adjusting priors based on the performance of previous tasks (Amit and Meir, 2018). In contrast to these approaches, we aim to define a suitable prior distribution entirely *a priori*. We acknowledge that our choice to impose GP (or hierarchical GP) priors on neural networks is essentially heuristic: there is no particular theory that necessarily claims superiority for this kind of prior distributions. In some applications, it could be preferable to use priors that are tailored to certain kinds of data or architectures, such the *deep weight prior* (Atanov et al., 2019). However, we are encouraged by the empirical success and the interpretability of GP models, and we seek to investigate their suitability as BNN priors on a wide range of regression and classification problems.

Our work is most closely related to a family of works that attempt to map GP priors to BNNS. Flam-Shepherd et al., 2017 propose to minimize the KL between the BNN prior and some desired GP. As there is no analytical form for this KL, the authors rely on approximations based on moment matching and projections on the observation space. This limitation was later addressed (Flam-Shepherd et al., 2018) by means of a hypernetwork (Ha et al., 2017), which generates the weight parameters of the original BNN; the hypernetwork parameters were trained so that BNN fit the samples of a GP. In our work, we also pursue to minimize a sample-based distance between the BNN prior and some desired GP, but we avoid the difficulties in working with the KL divergence, as its evaluation is challenging due to the empirical entropy term. To the best of our knowledge, the Wasserstein distance scheme we propose is novel, and it demonstrates satisfactory convergence for compatible classes of GPS and BNNS.

5.3 IMPOSING GAUSSIAN PROCESS PRIORS IN BAYESIAN NEURAL NETWORKS

The equivalence between function-space view and weight-space view of linear models, like Bayesian linear regression and GPS (Rasmussen and Williams, 2005), is a straightforward application of Gaussian identities, but it allows us to seamlessly switch point of view accordingly to which characteristics of the model we are willing to observe or impose. We would like to leverage this equivalence also for BNNS but the nonlinear nature of such models makes it analytically intractable (or impossible, for non-invertible activation functions). We argue that for BNNS—and Bayesian deep learning models, in general—starting from a prior over the weights is not ideal, given the impossibility of interpreting its effect on the family of functions that the model can represent. We therefore rely on an optimization-based procedure to impose functional priors on BNNS using the Wasserstein distance as a similarity metric between such distributions, as described next.

5.3.1 A quick introduction to the Wasserstein distance

The concept of distance between probability measures is central to this work, as we frame the problem of imposing a GP prior on a BNN as a distance minimization problem. We present some known results on the Wasserstein distance that will be used in the sections that follow. Given two Borel's probability measures $\pi(\mathbf{x})$ and $\nu(\mathbf{y})$ defined on the Polish space $\mathcal{X}$ and $\mathcal{Y}$ (i.e. any complete separable metric space), the generic formulation of the p-Wasserstein distance is defined as follows:

$$W_p(\pi,\nu) = \left(\inf_{\gamma\in\Gamma(\pi,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} D(\mathbf{x},\mathbf{y})^p \gamma(\mathbf{x},\mathbf{y})\, d\mathbf{x}\, d\mathbf{y}\right)^{1/p}, \tag{5.1}$$

where $D(\mathbf{x},\mathbf{y})$ is a proper distance metric between two points $\mathbf{x}$ and $\mathbf{y}$ in the space $\mathcal{X}\times\mathcal{Y}$ and $\Gamma(\pi,\nu)$ is the set of functionals of all possible joint densities γ whose marginals are π and ν.

When the spaces of $\mathbf{x}$ and $\mathbf{y}$ coincide (i.e. $\mathbf{x},\mathbf{y}\in\mathcal{X}\subseteq\mathbb{R}^d$), with $D(\mathbf{x},\mathbf{y})$ being the Euclidian norm distance, the Wasserstein-1 distance (also known in the literature as Earth-Mover distance) takes the following shape,

$$W_1(\pi,\nu) = \inf_{\gamma\in\Gamma(\pi,\nu)} \int_{\mathcal{X}\times\mathcal{X}} \|\mathbf{x}-\mathbf{y}\| \gamma(\mathbf{x},\mathbf{y})\, d\mathbf{x}\, d\mathbf{y}\,. \tag{5.2}$$

With the exception of few cases where the solution is available analytically (e.g. π and ν being Gaussians), solving Equation (5.2) directly or via optimization is intractable. On the other hand, the Wasserstein distance defined in Equation (5.2) admits the following dual form (Kantorovich, 1942; Kantorovich, 1948),

$$\begin{aligned} W_1(\pi,\nu) &= \sup_{\|\phi\|_L\leqslant 1} \left[\int \phi(\mathbf{x})\pi(\mathbf{x})d\mathbf{x} - \int \phi(\mathbf{y})\nu(\mathbf{y})d\mathbf{y}\right] \\ &= \sup_{\|\phi\|_L\leqslant 1} \mathbb{E}_\pi\phi(\mathbf{x}) - \mathbb{E}_\nu\phi(\mathbf{x})\,, \end{aligned} \tag{5.3}$$

where ϕ is a 1-Lipschitz continuous function defined on $\mathcal{X}\to\mathbb{R}$. This is effectively a functional maximization over ϕ on the difference two expectations of ϕ under π and ν. A revised proof of this dual form by Villani (2003) is available in the Appendix.

Following recent literature (Goodfellow et al., 2014; Arjovsky et al., 2017), we parameterize the Lipschitz function by a neural network with parameters θ. In this setup, the supremum is replaced with a maximization problem with respect to the parameters θ. In order to enforce the Lipschitz constraint on ϕ_θ, Arjovsky et al. (2017) propose to clip the weights θ to lie within a compact space $[-c, c]$ such that all function $\phi_\theta(\cdot)$ will be K-Lipschitz, with K usually unknown. This approach usually biases the resulting $\phi_\theta(\cdot)$ towards simple functions. Based on the fact that a differentiable function is 1-Lipschitz if and only if the norm of its gradient is at most one everywhere, Gulrajani et al. (2017) propose to constrain the gradient norm of the output of the Lipschitz function $\phi_\theta(\cdot)$ with respect to its input. More specifically, the loss of the Lipschitz function is augmented by a regularization term

$$\max_{\theta}\left[\mathbb{E}_{\pi(\mathbf{x})}\phi(\mathbf{x};\theta)-\mathbb{E}_{\nu(\mathbf{x})}\phi(\mathbf{x};\theta)+\lambda\mathbb{E}_{p(\hat{\mathbf{x}})}\left[\left(\|\nabla_{\hat{\mathbf{x}}}\phi(\hat{\mathbf{x}})\|_2-1\right)^2\right]\right] \tag{5.4}$$

Here $p(\hat{\mathbf{x}})$ is the distribution of $\hat{\mathbf{x}}=\varepsilon\tilde{\mathbf{x}}_\pi+(1-\varepsilon)\tilde{\mathbf{x}}_\nu$ for $\varepsilon\sim\mathcal{U}[0,1]$ and $\tilde{\mathbf{x}}_\pi\sim\pi(\mathbf{x})$, $\tilde{\mathbf{x}}_\nu\sim\nu(\mathbf{x})$ being the sample functions from the two marginals and λ the penalty coefficient.

5.3.2 Using the Wasserstein distance to impose the GP behaviour in BNN

Let now go back to our original problem of optimizing priors for BNNs. Assume a prior distribution $p(\mathbf{w};\psi)$ on the weights of a BNN, where ψ is a set of parameters that determine the prior (e.g. for a Gaussian prior, $\psi=\{\mu,\sigma\}$; we discuss more options on the parametrization of BNN priors in the section that follows). This prior over weights induces a prior distribution over functions:

$$p_{nn}(\mathbf{f};\psi)=\int p(\mathbf{f}\,|\,\mathbf{w})p(\mathbf{w};\psi)\,d\mathbf{w}, \tag{5.5}$$

where $p(\mathbf{f}\,|\,\mathbf{w})$ is deterministically defined by the network architecture.

We consider as our target distribution a GP defined as $p_{gp}(\mathbf{f}\,|\,\mathbf{0},\mathbf{K})$, where $\mathbf{K}$ is the covariance matrix obtained by computing the kernel function κ for each pair of datapoints $\{\mathbf{x}_i,\mathbf{x}_j\}$. We aim at matching these two stochastic processes at a finite number of measurement points $\mathbf{X}_\mathcal{M}\overset{\text{def}}{=}[\mathbf{x}_1,\dots,\mathbf{x}_M]^\top$. To achieve this, we propose a sample-based approach using the 1-Wasserstein distance as objective:

$$\min_{\psi}\max_{\theta}\left[\mathbb{E}_{p_{gp}}[\phi_\theta(\mathbf{f}_\mathcal{M})]-\mathbb{E}_{p_{nn}}[\phi_\theta(\mathbf{f}_\mathcal{M})]+\lambda\mathbb{E}_{p_{\hat{\mathbf{f}}}}\left[\left(\left\|\nabla_{\hat{\mathbf{f}}}\phi(\hat{\mathbf{f}})\right\|_2-1\right)^2\right]\right]. \tag{5.6}$$

where $\mathbf{f}_\mathcal{M}$ denotes the set of random variables associated with the inputs at $\mathbf{X}_\mathcal{M}$, and ϕ_θ is 1-Lipschitz function.

Regarding the optimization of the θ and ψ parameters we alternate between $n_{\text{Lipschitz}}$ steps of maximizing $\mathcal{L}$ with respect to the Lipschitz function's parameters θ and one step of minimizing the Wasserstein distance with respect to the prior's

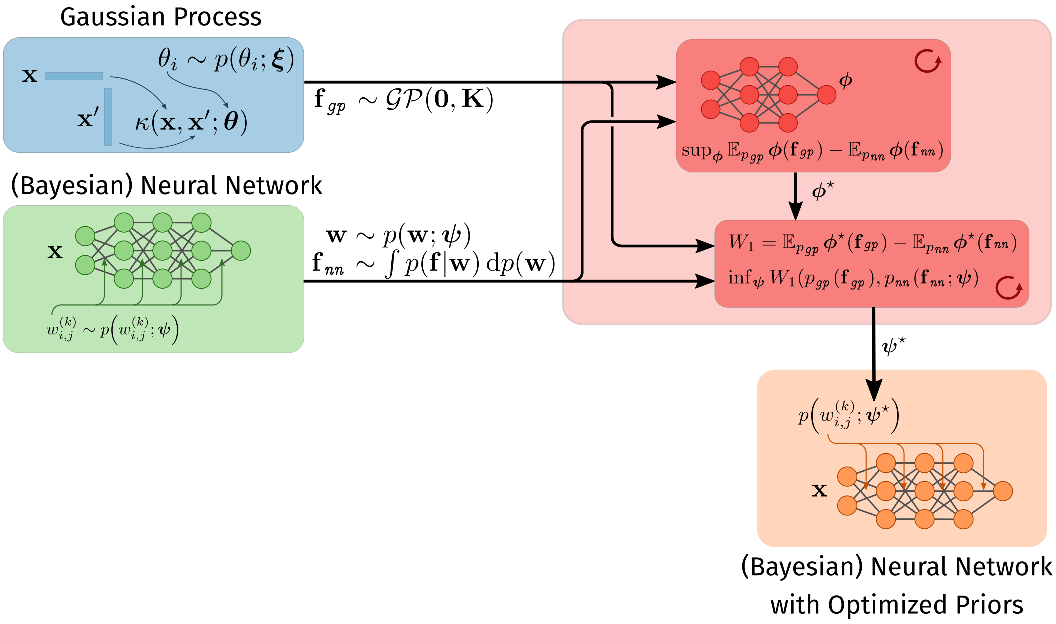

Gaussian Process
$\mathbf{x}$
$\theta_i \sim p(\theta_i; \boldsymbol{\xi})$
$\mathbf{x}'$
$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$
$\mathbf{f}_{gp} \sim \mathcal{GP}(\mathbf{0}, \mathbf{K})$
(Bayesian) Neural Network
$\mathbf{x}$
$w_{i,j}^{(k)} \sim p\left(w_{i,j}^{(k)}; \psi\right)$
$\mathbf{w} \sim p(\mathbf{w}; \psi)$
$\mathbf{f}_{nn} \sim \int p(\mathbf{f}|\mathbf{w})\, \mathrm{d}p(\mathbf{w})$
ϕ
$\sup_\phi \mathbb{E}_{p_{gp}} \phi(\mathbf{f}_{gp}) - \mathbb{E}_{p_{nn}} \phi(\mathbf{f}_{nn})$
$\phi^\star$
$W_1 = \mathbb{E}_{p_{gp}} \phi^\star(\mathbf{f}_{gp}) - \mathbb{E}_{p_{nn}} \phi^\star(\mathbf{f}_{nn})$
$\inf_\psi W_1(p_{gp}(\mathbf{f}_{gp}), p_{nn}(\mathbf{f}_{nn}; \psi))$
$\psi^\star$
$p\left(w_{i,j}^{(k)}; \psi^\star\right)$
$\mathbf{x}$
(Bayesian) Neural Network
with Optimized Priors

Figure 5.4: Schematic representation of the process of imposing GP priors on BNNs via Wasserstein distance minimization.

parameters ψ. We therefore use two independent optimizers for θ and ψ. Figure 5.4 offers a high-level schematic representation of the proposed procedure. Given samples from two stochastic processes, the Wasserstein distance is estimated by considering the inner maximization of Equation (5.6), resulting in an optimal ϕ^*. This inner optimization step is repeated for every step of the outer optimization loop. Notice that the objective is fully sample-based. As a result, it is not necessary to know the closed-form of the marginal density $p_{nn}(f; \psi)$. One may consider any stochastic process as a target prior over functions, as long as we can draw samples from it.

5.3.3 Prior Parameterization for Neural Networks

In the previous section, we have treated the parameters of a BNN prior $p_{nn}(f; \psi)$ in a rather abstract manner. Now we explore three different parametrizations of increasing complexity. The only two requirements needed to design a new parametrization are (1) to be able to generate samples and (2) to compute the log-density at any point; the latter is required to be able to draw samples from the posterior over model parameters using most MCMC sampling methods, such as Stochastic Gradient Hamiltonian Monte Carlo (SGHMC).

GAUSSIAN PRIOR. We consider a layer-wise factorization with two independent zero-mean Gaussian distributions for weights and biases. The parameters to adjust are $\psi = \{\sigma^2_{l_w}, \sigma^2_{l_b}\}_{l=1}^{L}$, where $\sigma^2_{l_w}$ is the prior variance shared across all weights in layer l, and $\sigma^2_{l_b}$ is the respective variance for the bias parameters. For any weight and bias entries $w_l, b_l \in \boldsymbol{w}_l$ of the l-th layer, the prior is:

$$p(w_l) = \mathcal{N}\left(w_l;\ 0, \sigma^2_{l_w}\right) \quad \text{and} \quad p(b_l) = \mathcal{N}\left(b_l;\ 0, \sigma^2_{l_b}\right)$$

We refer to this case as the GP-*induced* BNN *prior with Gaussian weights* (GPi-G). Although this simple approach assumes a Gaussian prior on the parameters, in many cases it is sufficient to capture the target GP-based functional priors.

HIERARCHICAL PRIOR. A more flexible family of priors for BNNs considers a hierarchical structure where the network parameters follow a conditionally Gaussian distribution, and the prior variance for each layer follows an Inverse-Gamma distribution. For the weight and bias variances we have:

$$\sigma^2_{l_w} \sim \Gamma^{-1}(\alpha_{l_w}, \beta_{l_w}) \quad \text{and} \quad \sigma^2_{l_b} \sim \Gamma^{-1}(\alpha_{l_b}, \beta_{l_b})$$

In this case, we have $\psi = \{\alpha_{l_w}, \beta_{l_w}, \alpha_{l_b}, \beta_{l_b}\}_{l=1}^{L}$, where $\alpha_{l_w}, \beta_{l_w}, \alpha_{l_b}, \beta_{l_b}$ denote the shape and rate parameters of the Inverse-Gamma distribution for the weight and biases correspondingly for layer l. The conditionally Gaussian prior over the network parameters is given as in the previous section. We refer to this parametrization as the GP-*induced* BNN *prior with Hierarchically-distributed weights* (GPi-H).

BEYOND GAUSSIANS WITH NORMALIZING FLOWS. Finally, we also consider normalizing flows (NFs) as a family of much more flexible distributions. By considering an invertible, continuous and differentiable function $t : \mathbb{R}^{D_l} \to \mathbb{R}^{D_l}$, where D_l is the number of parameters for l-th layer, a NF is constructed as a sequence of K of such transformations $\mathcal{T}_K = \{t_1, \ldots, t_K\}$ of a simple known distribution (e.g. Gaussian). Sampling from such distribution is as simple as sampling from the initial distribution and then apply the set of transformation $\mathcal{T}_K$. Given an initial distribution $p_0(\boldsymbol{w}_l)$, by denoting $p(\mathcal{T}_K(\boldsymbol{w}_l))$ the final distribution, its log-density can be analytically computed by taking into account to Jacobian of the transformations as follows,

$$\log p(\mathcal{T}_K(\boldsymbol{w}_l)) = \log p_0(\boldsymbol{w}_l) - \sum_{k=1}^{K} \log \left| \det \frac{\partial t_k(\boldsymbol{w}_{l_{k-1}})}{\partial \boldsymbol{w}_{l_{k-1}}} \right|, \tag{5.7}$$

where $\boldsymbol{w}_{l_{k-1}} = (t_{k-1} \circ \ldots \circ t_2 \circ t_1)(\boldsymbol{w}_l)$ for $k > 1$, and $\boldsymbol{w}_{l_0} = \boldsymbol{w}_l$.

We shall refer to this class of BNN priors as the GP*-induced* BNN *prior, parametrized by normalizing flows* (GPi-NF). We note that NFs are typically used differently in the literature; while previous works showed how to use this distributions for better approximation of the posterior in variational inference (D. Rezende and Mohamed, 2015; Kingma, Salimans, et al., 2016; Louizos and Welling, 2017) or for parametric density estimation (e.g. Grover et al., 2018), or for enlarging the flexibility of a prior for variational autoencoders (VAEs) (e.g. X. Chen et al., 2017), as far as we are aware this is the first time that NFs are used to characterize a prior distribution for BNNs.

A simple setup consists in setting the initial distribution $p_0(\boldsymbol{w}_l)$ to a fully-factorized Gaussian $\mathcal{N}(\boldsymbol{w}_l \,|\, \mathbf{0}, \sigma_l^2 \mathbf{I})$ and then employing a sequence of *planar flows* (D. Rezende and Mohamed, 2015), each defined as

$$t_k(\boldsymbol{w}_{l_{k-1}}) = \boldsymbol{w}_{l_{k-1}} + \mathbf{u}_{l_k} h(\boldsymbol{\theta}_{l_k}^\top \boldsymbol{w}_{l_{k-1}} + b_{l_k}), \tag{5.8}$$

where $\mathbf{u}_{l_k} \in \mathbb{R}^{D_l}, \boldsymbol{\theta}_{l_k} \in \mathbb{R}^{D_l}, b_{l_k} \in \mathbb{R}$ are trainable parameters, and $h(\cdot) = \tanh(\cdot)$. The log-determinant of the Jacobian of t_k is

$$\log \left| \det \frac{\partial t_k(\boldsymbol{w}_{l_{k-1}})}{\partial \boldsymbol{w}_{l_{k-1}}} \right| = \log \left| 1 + \mathbf{u}_{l_k}^\top \boldsymbol{\theta}_{l_k} h'(\boldsymbol{\theta}_{l_k}^\top \boldsymbol{w}_{l_{k-1}} + b_{l_k}) \right|. \tag{5.9}$$

Thus for the l-th BNN layer, the parameters to optimize are $\boldsymbol{\psi}_l = \{\sigma_l^2\} \bigcup \{\mathbf{u}_{l_k}, \boldsymbol{\theta}_{l_k}, b_{l_k}\}_{k=1}^{K}$.

5.4 EMPIRICAL EVALUATION

In the following experiments, we consider two fixed priors: (1) fixed Gaussian (FG) prior, $\mathcal{N}(0,1)$; (2) fixed hierarchical (FH) prior where the prior variance for each layer is sampled from an Inverse-Gamma distribution, $\Gamma^{-1}(1,1)$ (Springenberg et al., 2016); and two GP-induced neural network (NN) priors, namely: (3) GP-induced Gaussian (GPi-G) prior, (4) GP-induced hierarchical (GPi-H) prior. Since

Table 5.1: Glossary of methods used in the experimental campaign. Here, $p(\mathbf{f}) = \int p(\mathbf{f} \mid \boldsymbol{w})\, dp(\boldsymbol{w})$ denotes the induced prior over functions; $\Gamma^{-1}(\alpha, \beta)$ denotes the Inverse-Gamma distribution with shape α, and rate β; $\mathcal{NF}(\mathcal{T}_K)$ indicates a normalizing flow distribution constructed from a sequence of K invertible transformations $\mathcal{T}$; $\widehat{\sigma}^2$, and $(\widehat{\alpha}, \widehat{\beta})$ denote the optimized parameters for the GPi-G and GPi-H priors, respectively. $\widehat{\kappa}$ corresponds to optimized kernel parameters, while $\widehat{\sigma}^2_{\text{LA}}$ shows that the parameters are optimized on the Laplace approximation of the marginal likelihood. References are [a] for Wenzel et al. (2020), [b] for Springenberg et al. (2016), [c] for Lakshminarayanan et al. (2017).

	Priors			**Inference**	
Name	$p(\sigma^2)$	$p(\boldsymbol{w} \mid \sigma^2)$	$p(\mathbf{f})$		**Reference**
Fixed Gaussian (FG) prior	–	$\mathcal{N}(0, \sigma^2\mathbf{I}) \rightarrow$	?	SGHMC	
Fixed hierarchical (FH) prior	$\Gamma^{-1}(\alpha, \beta) \rightarrow$	$\mathcal{N}(0, \sigma^2\mathbf{I}) \rightarrow$	?	SGHMC + Gibbs	[b]
Fixed Gaussian prior and TS (FG+TS)	–	$\mathcal{N}(0, \sigma^2\mathbf{I}) \rightarrow$	?	Tempered SGHMC	[a]
Deep ensemble	–	?	?	Ensemble	[c]
GP-induced Gaussian (GPi-G) prior	–	$\mathcal{N}(0, \widehat{\sigma}^2\mathbf{I}) \leftarrow$	$\mathcal{GP}(0, \kappa)$	SGHMC	
GP-induced hierarchical (GPi-H) prior	$\Gamma^{-1}(\widehat{\alpha}, \widehat{\beta}) \leftarrow$	$\mathcal{N}(0, \sigma^2\mathbf{I}) \leftarrow$	$\mathcal{GP}(0, \kappa)$	SGHMC + Gibbs	
GP-induced norm. flow (GPi-NF) prior	–	$\mathcal{NF}(\mathcal{T}_K) \leftarrow$	$\mathcal{GP}(0, \kappa)$	SGHMC	

the computational cost of the GPi-NF prior is high, we don't consider this prior for large scale models.

We additionally compare BNNs against Deep Ensemble (Lakshminarayanan et al., 2017), arguably one of the state-of-the-art approaches for uncertainty estimation in deep learning (Ashukha et al., 2020; Ovadia et al., 2019). This non-Bayesian method combines solutions that maximize the predictive log-likelihood for multiple neural networks trained with different initializations. We employ an ensemble of 5 neural networks in all experiments. Following Lakshminarayanan et al. (2017), we use Adam optimizer (Kingma and Ba, 2015) to train the individual networks. Furthermore, we compare the results obtained by sampling from the posterior obtained with GP-induced priors against "tempered" posterior (Wenzel et al., 2020) that uses the FG prior and temperature scaling; we refer to this approach as FG+TS.

Finally, Table 6.1 summaries an overview of methods considered in the experiments.

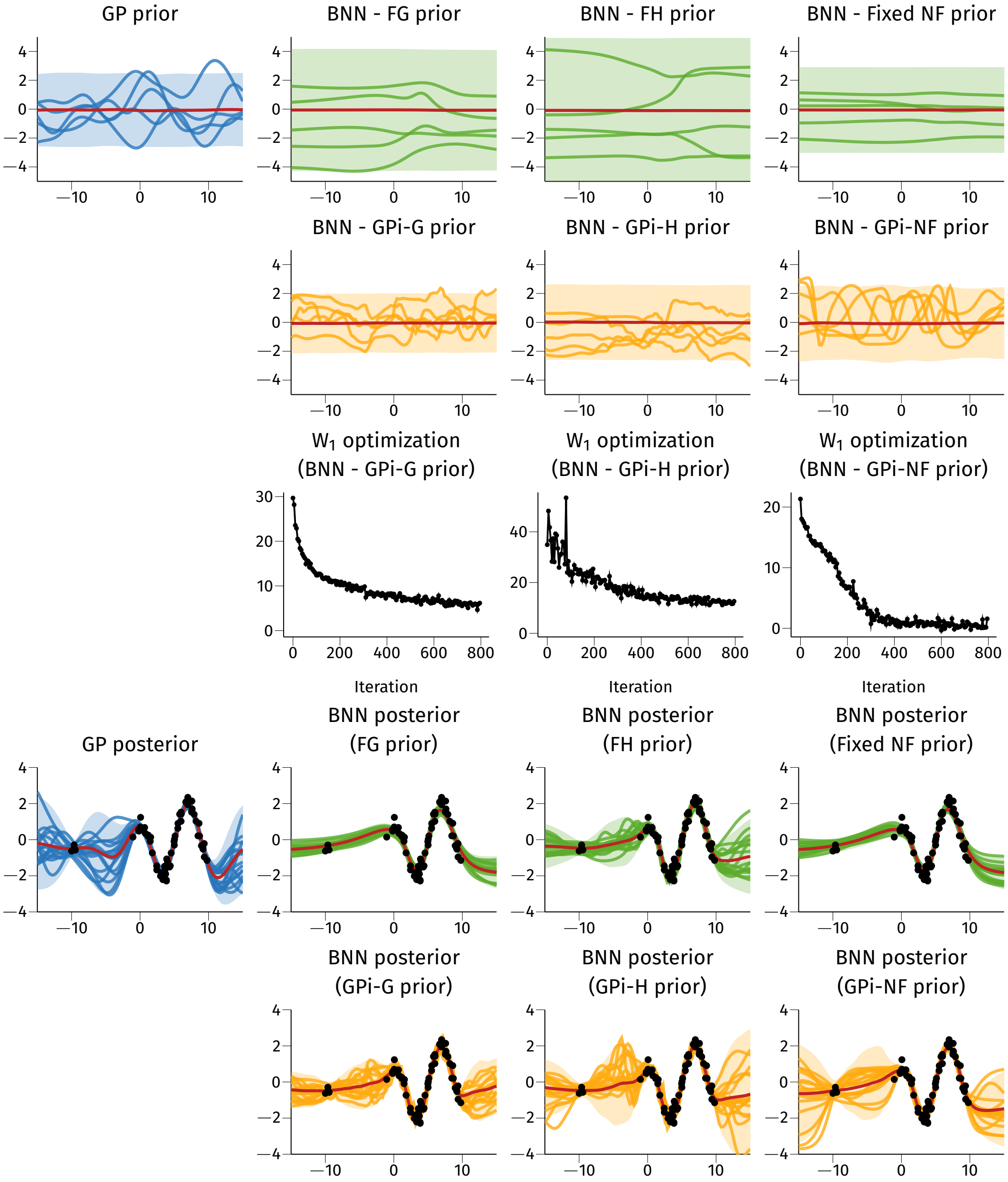


Figure 5.5: Visualization of one-dimensional regression example with a three hidden-layer multilayer perceptron (MLP). The first two rows illustrate the prior sample and distributions, whereas the last two rows show the corresponding posterior distributions. The means and the 95% credible intervals are represented by red lines and shaded areas, respectively. The middle row shows progressions of the prior optimization.

5.4.1 Visualization on a 1D regression synthetic dataset

The dataset used is built as follows: (1) we uniformly sample 64 input locations x in the interval $[-10, 10]$; (2) we rearrange the locations on a defined interval to generate a gap in the dataset; (3) we sample a function f from the GP prior ($l = 0.6, \alpha = 1$) computed at locations x; (4) we corrupt the targets with i.i.d. Gaussian noise ($\sigma_\epsilon^2 = 0.1$). In this example, we consider a three hidden-layer MLP. Figure 5.5 shows all the results. The first two rows illustrate the different choice of priors. For the Wasserstein-based functional priors (GPi-G, GPi-H, GPi-NF), the third row shows the convergence of the optimization procedure. Finally, the last two rows represent the posterior collected by running SGHMC with the corresponding priors.

From the analysis of these plots, we clearly see the benefit of placing a prior on the functions rather than on the parameters. First, the Wasserstein distance plots show satisfactory convergence, with the normalizing flow prior closely matching the GP prior. Second, as expected, the posteriors exhibit similar behavior according to the possible solutions realizable from the prior: classic priors tend to yield degenerate functions resulting in overconfidence in regions without data, while our GP-based priors (GPi-G, GPi-H, GPi-NF) retain information regarding lengthscale and amplitude.

5.4.2 Comparison for Bayesian convolutional neural networks

We proceed with the experimental campaign on the CIFAR10 benchmark (Krizhevsky and G. Hinton, 2009) with a number of popular CNN architectures: LeNet (LeCun, Bottou, et al., 1998), VGG (Simonyan and Zisserman, 2014) and PreResNet (K. He et al., 2016).

Regarding posterior inference, we implement SGHMC which, after a burn-in phase of 10,000 iterations, collects 200 samples with 10,000 simulation steps in between. For a fair comparison, we do not use techniques such as data augmentation or adversarial examples in any of the experiments. Regarding the target GP prior, we place a hyper-prior $\text{LogNormal}(\log 8, 0.3)$ for variance; whereas the hyper-prior for length-scale is $\text{LogNormal}(\log 512, 0.3)$. We use a mini-batch size of $N_s = 128$ and $N_{\mathcal{M}} = 32$ measurement points sampled from the empirical distribution of the training data regarding prior optimization.

Table 5.2 summarizes the results on the CIFAR10 test set with respect to accuracy and mean negative loglikelihood (MNLL). These results demonstrate the effectiveness of the GP-induced priors, as evidenced by the improvements in predictive

Table 5.2: Results for different convolutional neural networks on CIFAR10. Experimental comparison performed in B.-H. Tran et al. (2020) by the first author (Ba-Hien Tran).

Architecture	Method	Accuracy - % (↑)	NLL (↓)
LeNet	Deep Ensemble	71.13 ± 0.10	0.8548 ± 0.0010
	FG prior	74.65 ± 0.25	0.7482 ± 0.0025
	FG+TS	74.08 ± 0.24	0.7558 ± 0.0024
	GPi-G prior (**ours**)	75.15 ± 0.24	0.7360 ± 0.0024
	FH prior	75.22 ± 0.40	0.7209 ± 0.0040
	GPi-H prior (**ours**)	**76.51** ± **0.21**	**0.6952** ± **0.0021**
PreResNet	Deep Ensemble	87.77 ± 0.03	0.3927 ± 0.0003
	FG prior	85.34 ± 0.13	0.4975 ± 0.0013
	FG+TS	87.70 ± 0.11	0.3956 ± 0.0011
	GPi-G prior (**ours**)	86.86 ± 0.27	0.4286 ± 0.0027
	FH prior	87.26 ± 0.09	0.4086 ± 0.0009
	GPi-H prior (**ours**)	**88.20** ± **0.07**	**0.3808** ± **0.0007**
VGG	Deep Ensemble	81.96 ± 0.33	0.7759 ± 0.0033
	FG prior	81.47 ± 0.33	0.5808 ± 0.0033
	FG+TS	82.25 ± 0.15	0.5398 ± 0.0015
	GPi-G prior (**ours**)	83.34 ± 0.53	0.5176 ± 0.0053
	FH prior	86.03 ± 0.20	0.4345 ± 0.0020
	GPi-H prior (**ours**)	**87.03** ± **0.07**	**0.4127** ± **0.0007**

performance when using GPi-G and GPi-H priors compared to using FG and FH priors, respectively. Noticeably, the GPi-H prior offers the best performance with 76.51%, 87.03%, and 88.20% predictive accuracy on LeNet, VGG, and PreResNet, respectively. We observe that, for complex models (e.g., PreResNet and VGG), FG prior's results are improved by a large margin by tempering the posterior. This is in line with the results showed by Wenzel et al., 2020. By contrast, in the case of LeNet, the predictive performance dramatically degraded when using temperature scaling. This is clear evidence supporting the hypothesis that a "tempered" posterior is only useful for over-parameterized models. Instead, by using GP induced priors, we consistently obtain the best results in most cases.

5.4.3 Optimizing priors with data

Although we advocate for functional priors over BNNs, we acknowledge that a prior of this kind is essentially heuristic. A potentially more useful prior might be discovered by traditional means such as cross-validation (CV) or by running an empirical Bayes procedure (a.k.a. type-II maximum likelihood), which maximizes the marginal likelihood $p(\mathcal{D};\psi) = \int p(\mathcal{D}\,|\,\mathbf{w})p(\mathbf{w};\psi)\,d\mathbf{w}$ w.r.t. the prior parameters.

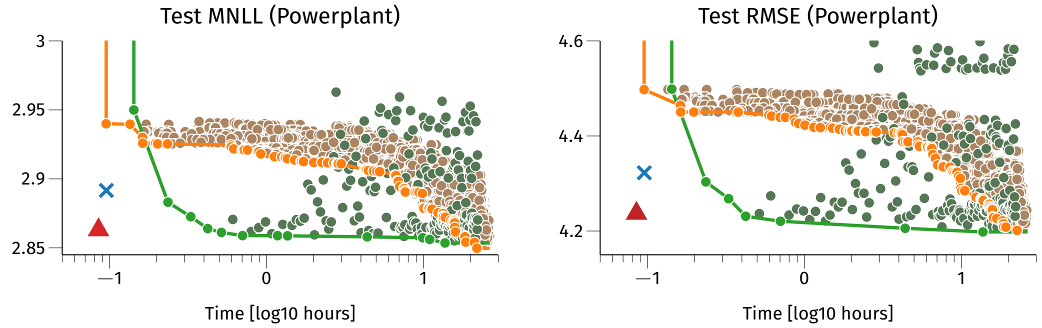


Figure 5.6: A timing comparison between imposing functional prior and cross-validation with grid-search (🟧) and using Bayesian optimization (🟩). In the plots, each ● corresponds to a run of a single configuration, while ─●─ highlights the Pareto front of the cross validation procedure. The figure also reports the $\mathcal{N}(0,1)$ prior as ×, while ▲ is our proposal of using functional prior (GPi-G).

These methods impose however significant challenges: (i) for CV, the number of hyperparameters that needs to be optimized becomes exponentially large as the complexity of the neural network grows and the more fine-grained grids are considered the higher the number of the combinations of parameters to explore becomes; (ii) for empirical Bayes, we need to compute the exact marginal likelihood, which is always intractable for BNNs, thus requiring additional approximations like variational inference (VI) or the Laplace approximation.

We consider a simple case of a BNN with one hidden layer only; by adopting the simple parameterization of § 5.3.3, we shall have four parameters to optimize in total (i.e. the weight and bias variances of the hidden and the output layer). In Figure 5.6, we demonstrate how our scheme behaves on a simple regression task (dataset is Powerplant) in comparison with a CV strategy featuring a grid size of 9 (for a total of 6561 configurations). To get results for the cross-validation procedure and to massively exploit all possible parallelization opportunities, we allocated a cloud platform with 16 workstations, for a total of 512 computing cores and 64 maximum parallel jobs. This required a bit more than one day, although the total CPU time approached 3 months. While grid-based routines are widely adopted by practitioners for cross-validation, we acknowledge that there are more efficient alternatives. To this extent, we also include Bayesian optimization (Močkus, 1975; Snoek et al., 2012; Nogueira, 2014), a classical method for black-box optimization which uses a Gaussian process as the surrogate function to be maximized (or minimized). As expected, CV indeed found marginally better configurations, but the amount of resources and time needed, even for such a small model, is orders of magnitude larger than what required by our scheme, making this procedure computationally infeasible for larger models, like CNNs. To reiterate and to put things into perspective, the Wasserstein-based functional prior could be run on a 4-cores laptop in a reasonable time, while CV required an entire rack of servers.

5.5 ANOTHER ROUTE FOR BAYESIAN OCCAM'S RAZOR

A common way to estimate hyper-parameters (i.e., prior parameters ψ) is to rely on the Bayesian Occam's razor (a.k.a. *empirical Bayes*), which dictates that the marginal likelihood $p_\psi(\mathbf{y})$ should be optimized with respect to ψ. There are countless examples where such simple procedure succeeds in practice (see, e.g., Rasmussen and Williams, 2005; Immer, Bauer, et al., 2021). We note however that marginal likelihood maximization for a large number of hyper-parameters can suffer from overfitting (Rasmussen and Williams, 2005; Ober et al., 2021). Nevertheless, we do not expect significant overfitting issues in our setting, as we focus on data that are characterized by a high level of structure (i.e. images). As we have seen, regular choices for the prior completely fail to capture the properties of such highly-structured outputs.

The marginal likelihood is obtained by marginalizing out the outputs $\mathbf{f}$ and the model parameters $\boldsymbol{w}$,

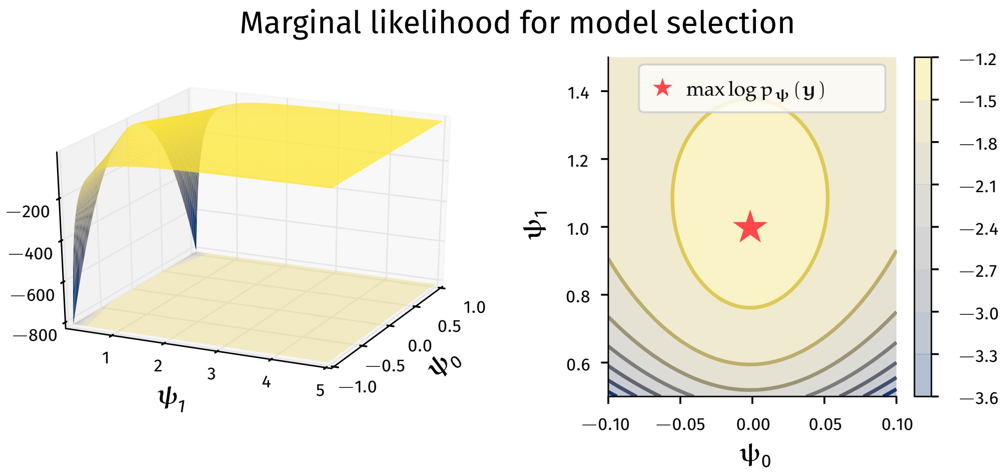
$$p_\psi(\mathbf{y}) = \int p(\mathbf{y} \,|\, \mathbf{f}) p_\psi(\mathbf{f}) \mathrm{d}\mathbf{f}, \tag{5.10}$$

where $p(\mathbf{y} \,|\, \mathbf{f})$ and $p_\psi(\mathbf{f})$ are the likelihood and the prior on functions, respectively. Unfortunately, in our context it is impossible to carry out this optimization due to the intractability of Equation (5.10).

Classic results in the statistics literature draw parallels between maximum likelihood estimation (MLE) and KL minimization (Akaike, 1973),

$$\arg\max_\psi \int \pi(\mathbf{y}) \log p_\psi(\mathbf{y}) \mathrm{d}\mathbf{y} = \arg\min_\psi \underbrace{\int \pi(\mathbf{y}) \log \frac{\pi(\mathbf{y})}{p_\psi(\mathbf{y})} \mathrm{d}\mathbf{y}}_{\mathrm{KL}[\pi(\mathbf{y}) \,\|\, p_\psi(\mathbf{y})]}, \tag{5.11}$$

where $\pi(\mathbf{y})$ is the true data distribution. This equivalence provides us with an interesting insight on an alternative view of marginal likelihood optimization as


Marginal likelihood for model selection

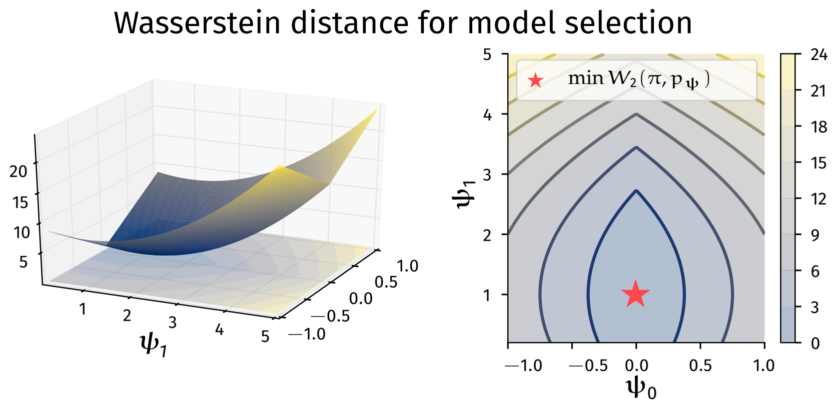

Wasserstein distance for model selection

Figure 5.7: Different loss functions for model selection share the same optimum, for linear models.

minimization of the divergence between the true data distribution and the marginal $p_\psi(y)$.

This alternative view allows one to obtain a viable optimization strategy that relies on an empirical estimate of the data distribution $\tilde{\pi}(y)$. This presents additional challenges however, as the empirical evaluation and optimization of KL divergences remains a well-known challenging problem (Flam-Shepherd et al., 2017). Although it is possible to evaluate KL (or any other f-divergence) empirically by leveraging results from convex analysis (Nguyen et al., 2010), we have opted to substitute KL with an alternative metric that is more convenient from a computational perspective. We thus employ the Wasserstein distance as a surrogate for KL divergence, so that we avoid the challenges of empirical KL estimation.

To summarize: (1) we would like to do prior selection by carrying out type-II MLE; (2) the MLE objective is analytically intractable but the connection with KL minimization allows us to (3) swap the divergence with the Wasserstein distance, yielding a practical framework for choosing priors.

5.5.1 The distributionally-sliced Wasserstein distance

Given the two probability measures π and p_ψ, both defined on $\mathbb{R}^D$ for simplicity, the p-Wasserstein distance between π and p_ψ is given by

$$W_p^p(\pi, p_\psi) = \inf_{\gamma \in \Gamma(\pi, p_\psi)} \int \|y - y'\|^p \gamma(y, y') dy dy', \tag{5.12}$$

where $\Gamma(\pi, p_\psi)$ is the set of all possible distributions $\gamma(y, y')$ such that the marginals are $\pi(y)$ and $p_\psi(y')$ (Villani, 2008). While usually analytically unavailable or computationally intractable, for $D = 1$ the distance has a simple closed form solution, that can be easily estimated using samples only (Kolouri et al., 2019).

The distributionally-sliced Wasserstein distance (DSWD) takes advantage of this result by projecting the estimation of distances for high-dimensional distributions into simpler estimation of multiple distances in one dimension. The projection is done using the Radon transform $\mathcal{R}$, an operator that maps a generic density function φ defined in $\mathbb{R}^D$ to the set of its integrals over hyperplanes in $\mathbb{R}^D$,

$$\mathcal{R}\varphi(t, \theta) := \int \varphi(\mathbf{r}) \delta(t - \mathbf{r}^\top \theta) d\mathbf{r}, \quad \forall t \in \mathbb{R}, \ \forall \theta \in \mathbb{S}^{D-1}, \tag{5.13}$$

where $\mathbb{S}^{D-1}$ is the unit sphere in $\mathbb{R}^D$ and $\delta(\cdot)$ is the Dirac delta (Helgason, 2010). Using the Radon transform, for a given direction (or *slice*) θ we can project the two densities π and p_ψ into one dimension and we can solve the optimal transport problem in this projected space. Furthermore, to avoid unnecessary computations, instead of considering all possible directions in $\mathbb{S}^{D-1}$, distributionally-sliced Wasserstein distance (DSWD) proposes to find the optimal probability measure of slices $\sigma(\theta)$ on the unit sphere $\mathbb{S}^{D-1}$,

$$DSW_p(\pi, p_\psi) := \sup_{\sigma \in \mathbb{M}_C} \left(\mathbb{E}_{\sigma(\theta)} W_p^p(\mathcal{R}\pi(t, \theta), \mathcal{R}p_\psi(t, \theta)) \right)^{1/p}, \tag{5.14}$$

where, for $C > 0$, $\mathbb{M}_C$ is the set of probability measures σ such that $\mathbb{E}_{\theta, \theta' \sim \sigma}[\theta^\top \theta'] \leqslant C$ (a constraint that aims to avoid directions to lie in only one small area). The direct computation of DSW_p in Equation (5.14) is still challenging but admits an equivalent dual form,

$$\sup_{h \in \mathcal{H}} \left\{ \left(\mathbb{E}_{\bar{\sigma}(\theta)} \left[W_p^p \left(\mathcal{R}\pi(t, h(\theta)), \mathcal{R}p_\psi(t, h(\theta)) \right) \right] \right)^{1/p} - \lambda_C \mathbb{E}_{\theta, \theta' \sim \bar{\sigma}} \left[\left| h(\theta)^\top h(\theta') \right| \right] \right\} + \lambda_C C, \tag{5.15}$$

where $\bar{\sigma}$ is a uniform distribution in $\mathbb{S}^{D-1}$, $\mathcal{H}$ is the set of functions $h : \mathbb{S}^{D-1} \to \mathbb{S}^{D-1}$ and λ_C is a regularization hyper-parameter. The formulation in Equation (5.15) is obtained by employing the Lagrangian duality theorem and by reparameterizing $\sigma(\theta)$ as push-forward transformation of a uniform measure in $\mathbb{S}^{D-1}$ via h. Now, by parameterizing h using a deep neural network with parameters $\boldsymbol{\phi}$, defined as $h_{\boldsymbol{\phi}}$, Equation (5.15) becomes an optimization problem with respect to the network parameters. The final step is to approximate the analytically intractable expectations with Monte Carlo integration,

$$\max_{\boldsymbol{\phi}} \left\{ \left[\frac{1}{K} \sum_{i=1}^{K} \left[W_p^p \left(\mathcal{R}\pi(t, h_{\boldsymbol{\phi}}(\theta_i)), \mathcal{R}p_\psi(t, h_{\boldsymbol{\phi}}(\theta_i)) \right) \right] \right]^{1/p} - \frac{\lambda_C}{K^2} \sum_{i,j=1}^{K} |h_{\boldsymbol{\phi}}(\theta_i)^\top h_{\boldsymbol{\phi}}(\theta_j)| \right\} + \lambda_C C,$$

with $\theta_i \sim \bar{\sigma}(\theta)$. Finally, we can use stochastic gradient methods to update $\boldsymbol{\phi}$ and then use the resulting optima for the estimation of the original distance.

5.5.2 Matching the marginal distribution to the data distribution via Wasserstein distance minimization

We aim at learning the prior parameters by optimizing the marginal $p_\psi(\mathbf{y})$ obtained after integrating out the weights from the joint $p_\psi(\mathbf{y}, \mathbf{w})$. The connection with *empirical Bayes* and KL minimization suggests that we can find the optimal $\psi^\star$ by minimizing the KL between the true data distribution $\pi(\mathbf{y})$ and the marginal $p_\psi(\mathbf{y})$. However, matching these two distributions is non-trivial due to their high dimensionality and the unavailability of their densities. To overcome this problem, we propose a sample-based approach using the distributional sliced 2-Wasserstein distance (Equation (5.15)) as objective:

$$\psi^\star = \arg\min_{\psi} \left[\mathrm{DSW}_2(p_\psi(\mathbf{y}), \pi(\mathbf{y})) \right]. \tag{5.16}$$

This objective function is flexible and does not require the closed-form of either $p_\psi(\mathbf{y})$ or $\pi(\mathbf{y})$. The only requirement is that we can draw samples from these two distributions. Note that we can sample from $p_\psi(y)$, by first computing f after sampling from $p_\psi(w)$ and then perturbing the generated f by sampling from the likelihood $p(y \mid f)$. In the next section, we will take a challenging application (unsupervised learning with an auto-encoding architecture) where this formulation of the prior learning problem will be implemented.

5.6 MODEL SELECTION FOR BAYESIAN AUTOENCODERS

The problem of learning useful representations of data that facilitate the solution of downstream tasks such as clustering, generative modeling and classification, is at the crux of the success of many machine learning applications (see, e.g., Bengio, Courville, et al., 2013, and references therein). From a plethora of potential solutions to this problem, unsupervised approaches based on autoencoders (Cottrell et al., 1989) are particularly appealing as, by definition, they do not require label information and have proved effective in tasks such as dimensionality reduction and information retrieval (G. E. Hinton and R. R. Salakhutdinov, 2006).

Autoencoders are neural network models composed of two parts, usually referred to as the encoder and the decoder. The encoder maps each input y_i to a set of lower-dimensional latent variables z_i. The decoder maps the latent variables z_i back to the observations y_i. The bottleneck introduced by the low-dimensional latent space is what characterizes the compression and representation learning capabilities of autoencoders. It is not surprising that these models have connections with principal component analysis, factor analysis and density networks (D. J. MacKay and Gibbs, 1999), and latent variable models (Lawrence, 2005).

In applications where quantification of uncertainty is a primary requirement or where data is scarce, it is important to carry out a Bayesian treatment of these models by specifying a prior distribution over their parameters, i.e., the weights of the encoder/decoder. However, estimating the posterior distribution over the parameters of these models, which we refer to as Bayesian auto-encoders (BAEs), is generally intractable and requires approximations. Furthermore, the need to specify priors for a large number of parameters, coupled with the fact that autoencoders are not generative models, has motivated the development of VAEs as an alternative that can overcome these limitations (Kingma and Welling, 2014). Indeed, VAEs have found tremendous success and have become one of the preferred methods in modern machine-learning applications (see, e.g., Kingma and Welling, 2019, and references therein).

To recap, three potential limitations of BAEs hinder their widespread applicability in order to achieve a similar or superior adoption to their variational counterpart: (i) lack of generative modeling capabilities; (ii) intractability of inference and (iii) difficulty of setting sensible priors over their parameters. In this work we revisit BAEs and deal with these limitations in a principled way. In particular, we address the first limitation in (i) by employing density estimation in the latent space. Furthermore, we deal with the second limitation in (ii) by exploiting recent advances in MCMC and, in particular, SGHMC (T. Chen et al., 2014). Finally, we believe that the third limitation (iii), which we refer to as the difficulty of carrying out *model selection*, requires a more detailed treatment because choosing sensible priors for Bayesian neural networks is an extremely difficult problem, and this is the main focus of this work.

5.6.1 Formalization of Bayesian Autoencoders

An auto-encoder (AE) is a neural network parameterized by a set of parameters $\boldsymbol{w}$, which transforms an unlabelled dataset, $\mathbf{y} \stackrel{\text{def}}{=} \{\mathbf{y}_n\}_{n=1}^N$, into a set of reconstructions $\hat{\mathbf{y}} \stackrel{\text{def}}{=} \{\mathbf{f}_n\}_{n=1}^N$, with $\mathbf{y}_n, \mathbf{f}_n \in \mathbb{R}^D$. An AE is composed of two components: (1) an encoder f_{enc} which maps an input sample $\mathbf{y}_n$ to a latent code $\mathbf{z}_n \in \mathbb{R}^K, K \ll D$; and (2) a decoder f_{dec} which maps the latent code to a reconstructed datapoint $\mathbf{f}_n$. In short, $\mathbf{f} = f(\mathbf{x}; \boldsymbol{w}) = (f_{\text{dec}} \circ f_{\text{enc}})(\mathbf{x})$, where we denote $\boldsymbol{w} := \{\boldsymbol{w}_{\text{enc}}, \boldsymbol{w}_{\text{dec}}\}$ the union of parameters of the encoder and decoder. The Bayesian treatment of AEs dictates that a prior distribution $p(\boldsymbol{w})$ is placed over all parameters of f_{enc} and f_{dec}, and that this prior knowledge is transformed into a posterior distribution by means of Bayes' theorem,

$$p(\boldsymbol{w} \,|\, \mathbf{y}) = \frac{p(\mathbf{y} \,|\, \boldsymbol{w}) p(\boldsymbol{w})}{p(\mathbf{y})}, \tag{5.17}$$

where $p(\mathbf{y} \,|\, \boldsymbol{w})$ is the conditional likelihood that factorizes as $p(\mathbf{y} \,|\, \boldsymbol{w}) = \prod_{n=1}^N p(\mathbf{y}_n \,|\, \boldsymbol{w})$. Note that each conditional likelihood term is determined by the model architecture, the choice of $\boldsymbol{w}$, and the input $\mathbf{x}_n$, but in order to keep the notation uncluttered, we write them simply as $p(\mathbf{y}_n \,|\, \boldsymbol{w})$.

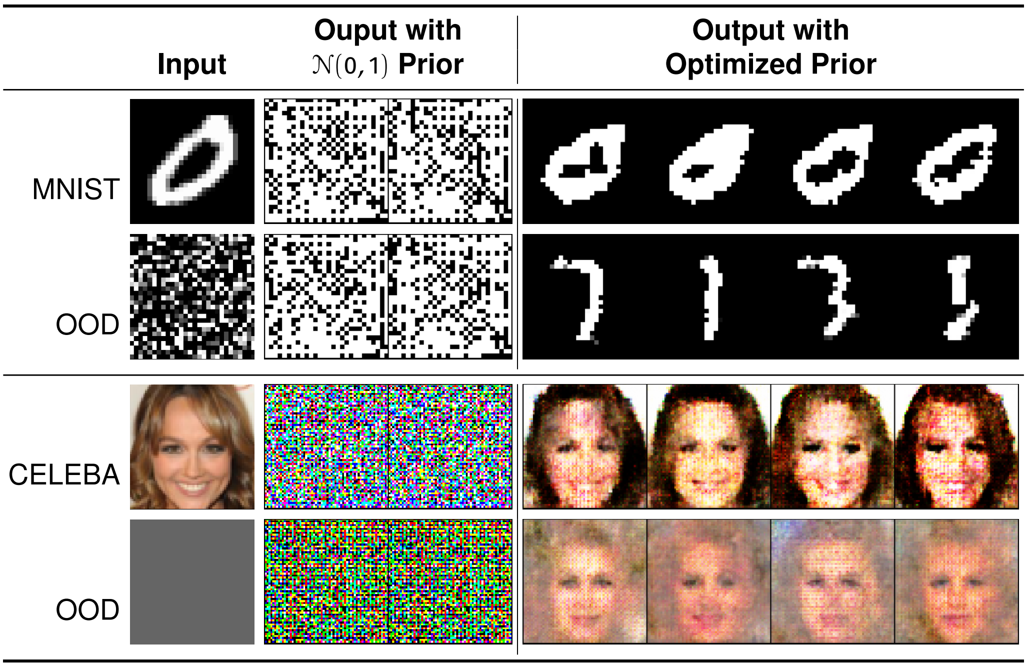

Input
Ouput with $\mathcal{N}(0,1)$ Prior
Output with Optimized Prior
MNIST
OOD
CELEBA
OOD

Figure 5.8: Realizations sampled from different priors given an input image. OOD stands for out-of-distribution.

LIKELIHOOD MODEL. In the Bayesian scheme, the prior and likelihood are both modeling choices. Before giving an in-depth treatment on priors for BAEs in the next section, we briefly discuss the likelihood, which can be chosen according to the type of data. In our experiments, we mainly investigate image datasets, where pixel values are normalized in the $[0,1]$ range. Therefore, we rely on the *continuous Bernoulli* distribution (Loaiza-Ganem and Cunningham, 2019):

$$p(\mathbf{y}_n \mid \mathbf{w}) = \prod_{i=1}^{D} K(\lambda_i)\lambda_i^{y_{n,i}}(1-\lambda_i)^{1-y_{n,i}} := p(\mathbf{y}_n \mid \mathbf{f}_n), \tag{5.18}$$

where $K(\lambda_i)$ is a properly defined normalization constant (Loaiza-Ganem and Cunningham, 2019) and $\lambda_i = f_i(\mathbf{x}_n; \mathbf{w}) = f_{n,i} \in [0,1]$ is the i-th output from the BAE given the input $\mathbf{x}_n$. We note that, as $\mathbf{f}_n$ depends deterministically on $\mathbf{w}$, we will use the above expression to refer to both $p(\mathbf{y}_n \mid \mathbf{w})$ and $p(\mathbf{y}_n \mid \mathbf{f}_n)$, where the latter term will be of crucial importance when we define the functional prior induced over the reconstruction.

5.6.2 The pathology of standard priors for BAEs and how to fix it

The choice of the prior is important for the Bayesian treatment of any model as it characterizes the hypothesis space (D. J. C. MacKay, 1992; Murray and Ghahramani, 2005). Specifically for BAEs, one should note that placing a prior on the parameters of the encoder and decoder has an implicit effect on the prior over the network output (i.e. the reconstruction). In addition, the highly nonlinear nature of these models implies that interpreting the effect of the architecture is theoretically intractable and practically challenging. Several works argue that a vague prior such as $\mathcal{N}(0,1)$ is good enough for some tasks and models, like classification with CNNs (A. G. Wilson and Izmailov, 2020b).

However, for BAEs this is not enough, as illustrated in Figure 5.8. The realizations obtained by sampling weights/biases from a $\mathcal{N}(0,1)$ prior indicate that this choice provides poor inductive bias. Meanwhile, by encoding better beliefs via an optimized prior, the samples can capture main characteristics intrinsic to the data, even when the model is fed with out-of-distribution inputs.

To overcome this problem, we propose a sample-based approach using the distributional sliced 2-Wasserstein distance in Equation (5.15) as objective:

$$\psi^{\star} = \arg\min_{\psi} \left[\mathrm{DSW}_2(p_{\psi}(\mathbf{y}), \pi(\mathbf{y})) \right]. \tag{5.19}$$

We will refer to the original paper (B. Tran et al., 2021) for the experimental evaluations, as mainly perfomed by the first author. As a summary, the experiments include a demonstration of the effect of our model selection strategy, by considering scenarios in the small-data regime where the prior might not be necessarily tuned on the training set. In this way we are able to impose inductive bias beyond what is available in the training data. The BAE with optimized priors performs better than the competing methods (VAEs and the BAE with standard prior) in the inference task for all training sizes, with slightly diminishing effect for larger sets, as expected. Also, this pattern is true for different latent dimensions, where regardless of the dimensionality of the latent space, BAEs with optimized priors deliver higher performances.

5.7 CONCLUDING REMARKS

Being able to perform Bayesian inference of neural networks represents a much sought-after objective to equip extremely flexible models with the capability of expressing uncertainty in a sound way (Neal, 1996). Recent advances in MCMC sampling enabling for efficient parameter space exploration, combined with minibatching (T. Chen et al., 2014), have turned this long-standing challenge into a concrete possibility. However despite these advances, there have been only few success stories involving the use of Bayesian inference techniques for neural networks (Osawa et al., 2019; R. Zhang et al., 2020; Izmailov et al., 2021). We attribute this to the difficulties in specifying sensible priors for thousands/millions of parameters, while being able to understand and control the effect of these choices in the behavior of their output functions (Duvenaud et al., 2014).

In this chapter we analyzed the problem of selection good priors from two different perspectives: functional priors with target stochastic processes and empirical Bayes with the Wasserstein distance as a proxy to the marginal likelihood.

The difficulty in reasoning about functional priors for neural networks made us consider, for the first work, the possibility to enforce these by minimizing their distance to tractable functional priors, effectively optimizing the priors over model parameters so as to reflect these functional specifications. We chose to consider Gaussian processes, as they are a natural and popular choice to construct functional priors, whereby the characteristics of prior functions are determined by the form and parameters of Gaussian process kernel/covariance functions. While previous works attempted this by using the KL divergence between the functional priors (Flam-Shepherd et al., 2017; Flam-Shepherd et al., 2018), the objective proves difficult to work with due to the need to estimate an entropy term based on samples, which is notoriously difficult. We proposed a novel objective based on the Wasserstein distance, and we showed that this objective offers a tractable and stable way to optimize the priors over model parameters. The attractive property of this objective is that it does not require a closed form for the target functional prior, as long as it is possible to obtain samples from it.

For the second work, we have reconsidered as a challenging benchmark the Bayesian treatment of autoencoders (AE) in light of recent advances in Bayesian neural networks. We have found that the main limitation of BAEs lies in the difficulty of specifying meaningful priors in the context of highly-structured data, which is ubiquitous in modern machine learning applications. Consequently, we have proposed to specify priors over the autoencoder weights by means of a novel optimization of prior hyper-parameters. Inspired by connections with marginal likelihood optimization, we derived a practical and efficient optimization framework, based on the minimization of the distributional sliced-Wasserstein distance between the distribution induced by the BAE and the data generating distribution. The resulting hyper-parameter optimization strategy leads to a novel way to perform model selection for BAEs. Note that, even if theoretically justified and empirically verified with extensive experimentation, our proposal for model selection still remains a *proxy* to the true marginal likelihood maximization. The DSWD formulation has nice properties of asymptotic convergence and computational tractability, but it may represent only one of the possible solutions. At the same time, we stress that the current literature does not cover this problem of BAEs at all, and we believe our approach is a considerable step towards the development of practical Bayesian methods for representation learning in modern applications characterized by large-scale structured data (including tabular and graph data, which are currently not covered).

6

REVISITING THE APPROXIMATIONS FOR SCALABLE (DEEP) GAUSSIAN PROCESSES

Variational inference techniques based on inducing variables provide an elegant framework for scalable posterior estimation in Gaussian process (GP) models. Besides enabling scalability, one of their main advantages over sparse approximations using direct marginal likelihood maximization is that they provide a robust alternative for point estimation of the inducing inputs, i.e. the location of the inducing variables. In this work we challenge the common wisdom that optimizing the inducing inputs in the variational framework yields optimal performance. We show that, by revisiting old model approximations such as the fully-independent training conditionals endowed with powerful sampling-based inference methods, treating both inducing locations and GP hyper-parameters in a Bayesian way can improve performance significantly. Based on stochastic gradient Hamiltonian Monte Carlo, we develop a fully Bayesian approach to scalable GP and deep GP models, and demonstrate its state-of-the-art performance through an extensive experimental campaign across several regression and classification problems.

6.1 SPARSE GAUSSIAN PROCESSES

Bayesian kernel machines based on GPs combine the modeling flexibility of kernel methods with the ability to carry out sound quantification of uncertainty (Rasmussen and Williams, 2005). Modeling and inference in GP models have evolved

Table 6.1: A summary of previous works on inference methods for GPs. θ, u, Z refer to the GP hyper-parameters, inducing variables and inducing inputs, respectively. (✗) indicates that variables are optimized.

		Inference			
	Model	θ	u	Z	Reference
	MCMC-GP	MCMC	-	-	Neal (1997) and Barber and Williams (1997)
(🟠)	SVGP	✗	VB	✗	Hensman, A. Matthews, et al. (2015)
(🟡)	FITC-SVGP	✗	VB (heterosk.)	✗	Titsias (2009b)
(🔴)	SGHMC-DGP	✗	MCMC	✗	Havasi et al. (2018)
(🟢)	IPVI-DGP	✗	IP	✗	H. Yu et al. (2019)
(🟣)	MCMC-SVGP	MCMC	MCMC	✗	Hensman, A. G. Matthews, et al. (2015)
(🔵)	BSGP	MCMC	MCMC	MCMC	This work

considerably over the last few years with key contributions in the direction of scalability to virtually any number of datapoints and generality within automatic differentiation frameworks (A. G. Matthews et al., 2017; Krauth et al., 2017). This has been possible thanks to the combination of stochastic variational inference techniques with representations based on inducing variables (Titsias, 2009b; Lazaro-Gredilla and Figueiras-Vidal, 2009; Hensman, Fusi, et al., 2013), random features (Rahimi and Recht, 2008; Cutajar et al., 2017; Gal and Ghahramani, 2016b), and structured approximations (A. Wilson and Nickisch, 2015; A. G. Wilson, Hu, R. R. Salakhutdinov, et al., 2016). These advancements have now made GPs attractive to a variety of applications and likelihoods (A. G. Matthews et al., 2017; Wilk et al., 2017; Bonilla et al., 2019).

In this work, we focus on the variationally sparse GP framework originally formulated by Titsias (2009b) and later developed by Hensman, Fusi, et al. (2013) and Hensman, A. Matthews, et al. (2015) to scale up to large datasets via stochastic optimization. In these formulations, the GP prior is augmented with inducing variables (drawn from the same prior) and their posterior is estimated via variational inference. In contrast, the location of the inducing variables, which we refer to as the inducing inputs, are simply optimized along with covariance hyper-parameters. In line with earlier evidence that Bayesian treatments of GPs are beneficial (Neal, 1997; Barber and Williams, 1997; Murray and R. P. Adams, 2010; Filippone and Girolami, 2014), posterior inference of the inducing variables *jointly* with covariance hyper-parameters has been shown to improve performance (Hensman, A. G. Matthews, et al., 2015).

Despite these significant insights with regards to the benefits of full Bayesian inference over latent variables in GP models, the common practice is to optimize the inducing inputs, even in very recent GP developments (Havasi et al., 2018; Shi

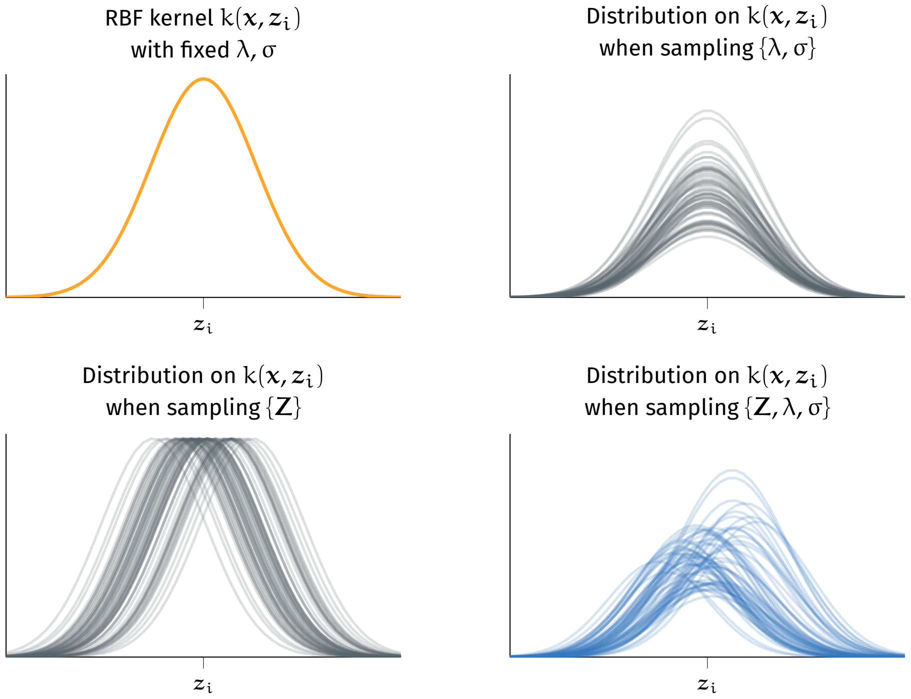


Figure 6.1: Representation of the induced distribution on the covariance function at location x when placing priors on different set of parameters.

et al., 2020; Giraldo and Álvarez, 2019). In fact, the original work of Titsias (2009b) advocates for a treatment of the inducing inputs as variational parameters to avoid overfitting. Furthermore, later work concludes that point estimation of the inducing inputs through optimization of the variational objective is an 'optimal' treatment (Hensman, A. G. Matthews, et al., 2015, §3). As we will see in § 6.2.1, the justification for inducing-input optimization in Hensman, A. G. Matthews, et al. (2015) relies on being able to optimize both the prior and the posterior, and therefore, contradicts the fundamental principles of Bayesian inference. We summarize previous works on inference methods for GPs in Table 6.1, which we will use for comparison in our experiments.

Thus, we revisit the role of the inducing inputs in GP models and their treatment as variational parameters or even hyper-parameters. Given their potential high dimensionality and that the typical number of inducing variables goes beyond hundreds/thousands (Shi et al., 2020), we argue that they should be treated simply as model variables and, therefore, having priors and carrying out efficient posterior inference over them is an important—although challenging—problem. An illustration of the richer modeling capabilities offered by treating inducing inputs in a Bayesian fashion is given in Figure 6.1.

CONTRIBUTIONS. Firstly, we challenge the common wisdom that optimizing the inducing inputs in the variational framework yields optimal performance. We show that, by revisiting old model approximations such as the fully independent training conditionals (FITC; see, e.g., Quiñonero-Candela and Rasmussen, 2005) endowed with powerful sampling-based inference methods, treating both inducing locations and GP hyper-parameters in a Bayesian way can improve performance significantly. We describe the conceptual justification and the mathematical details of our general formulation in § 6.2 and § 6.3. We then demonstrate that our approach yields state-of-the-art performance across a wide range of competitive benchmark methods, large-scale datasets and a variety of GP and deep GP models in § 6.4.

6.2 BAYESIAN SPARSE GAUSSIAN PROCESSES

We are interested in supervised learning problems with N input-label training pairs $\{\mathbf{X}, \mathbf{y}\} \overset{\text{def}}{=} \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where we consider a conditional likelihood $p(\mathbf{y} \mid \mathbf{f})$ and $\mathbf{f}$ is drawn from a zero-mean GP prior with covariance function $k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$ with hyper-parameters $\boldsymbol{\theta}$. Thus, we have that $p(\mathbf{f}) = \mathcal{N}(0, \mathbf{K}_{xx|\theta})$, where $\mathbf{K}_{xx|\theta}$ is the $N \times N$ covariance matrix obtained by evaluating $k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta})$ over all input pairs $\{\mathbf{x}_i, \mathbf{x}_j\}$. Inference in these types of models generally involves the costly $\mathcal{O}(N^3)$ operations to compute the inverse and log-determinant of the covariance matrix $\mathbf{K}_{xx|\theta}$.

FULL JOINT DISTRIBUTION OF SPARSE APPROXIMATIONS. Sparse GPs are a family of approximate models that address the scalability issue by introducing a set of M inducing variables $\mathbf{u} = (u_1, \ldots, u_M)$ at corresponding inducing inputs $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ such that $u_i = f(\mathbf{z}_i)$ (see, e.g., Quiñonero-Candela and Rasmussen, 2005). These inducing variables are assumed to be drawn from the same GP as the original process, yielding the joint prior $p(\mathbf{f}, \mathbf{u}) = p(\mathbf{u})p(\mathbf{f}|\mathbf{u})$. We consider a general formulation where we place priors $p_{\psi}(\boldsymbol{\theta})$ over covariance hyper-parameters and $p_{\xi}(\mathbf{Z})$ over inducing inputs with hyper-parameters ψ, ξ,

$$p(\boldsymbol{\theta}, \mathbf{Z}, \mathbf{u}, \mathbf{f}, \mathbf{y} \mid \mathbf{X}) = p_{\psi}(\boldsymbol{\theta}) p_{\xi}(\mathbf{Z}) p(\mathbf{u} \mid \mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{y} \mid \mathbf{f}), \tag{6.1}$$

where $p(\mathbf{u} \mid \mathbf{Z}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{zz|\theta})$, $p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{K}_{xz|\theta}\mathbf{K}_{zz|\theta}^{-1}\mathbf{u}, \mathbf{K}_{xx|\theta} - \mathbf{K}_{xz|\theta}\mathbf{K}_{zz|\theta}^{-1}\mathbf{K}_{xz|\theta}^{\top})$. The matrices $\mathbf{K}_{zz|\theta}, \mathbf{K}_{xz|\theta}$ denote the covariance matrices computed between points in $\mathbf{Z}$ and $\{\mathbf{X}, \mathbf{Z}\}$, respectively. We assume a factorized likelihood $p(\mathbf{y} \mid \mathbf{f}) = \prod_{n=1}^{N} p(y_n \mid f_n)$ and make no assumptions about the other distributions. In our formulation, approaches that do not consider priors over covariance hyper-parameters or inducing inputs correspond to improper uniform priors in Equation (6.1).

6.2.1 On scalable inference frameworks for GP models

Let $\mathbf{\Psi} \overset{\text{def}}{=} \{\mathbf{u}, \mathbf{Z}, \theta\}$ be the variables whose posterior we wish to infer. Our main object of interest is the log joint marginal obtained by integrating out the latent variables $\mathbf{f}$ in Equation (6.1), i.e., $\log p(\mathbf{y}, \mathbf{\Psi} \mid \mathbf{X}) = \log \int_{\mathbf{f}} p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{\Psi}, \mathbf{X}) d\mathbf{f} + \log p(\mathbf{\Psi})$. In particular, we are interested in discussing approximations to this that decompose over observations, allowing the use of stochastic optimization techniques to scale up to large datasets. In the literature of sparse GPs (see, e.g., Bauer et al., 2016; T. D. Bui et al., 2017), two of the most influential methods for carrying out inference on such models are based on the variational free energy (VFE) framework (Titsias, 2009b) and the fully independent training conditional (FITC) framework (E. Snelson and Ghahramani, 2006).

VFE APPROXIMATIONS. The key innovation in Titsias (2009b) is the definition of the approximate posterior $q(\mathbf{f}, \mathbf{u}) \overset{\text{def}}{=} q(\mathbf{u}) p(\mathbf{f} \mid \mathbf{\Psi}, \mathbf{X})$, where $q(\mathbf{u})$ is the variational posterior, which yields the evidence lower bound (ELBO)

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}, \theta) \geqslant -\mathrm{KL}\left[q(\mathbf{u}) \parallel p(\mathbf{u} \mid \mathbf{Z}, \theta)\right] + \mathbb{E}_{q(\mathbf{f}, \mathbf{u})} \log p(\mathbf{y} \mid \mathbf{f}) \overset{\text{def}}{=} \mathcal{L}_{\text{ELBO}} \,. \tag{6.2}$$

We note that this approach does not incorporate priors over inducing inputs or hyper-parameters. Inference involves constraining $q(\mathbf{u})$ to a parametric form and finding its parameters to optimize the ELBO. Titsias (2009b) correctly argues that in the regression setting the variational approach to inducing variable approximations should be more robust to overfitting than a direct marginal likelihood maximization approach of traditional approximate models such as those described in Quiñonero-Candela and Rasmussen (2005). Indeed, if inducing inputs $\mathbf{Z}$ are optimized then the resulting ELBO provides an additional regularization term (see Titsias, 2009b, §3 for details). However, as we shall see later, the benefits of being Bayesian about the inducing inputs and estimating their posterior distribution can be superior to those obtained by this regularization.

Restricting the form of $q(\mathbf{u})$ is suboptimal, and Hensman, A. G. Matthews, et al. (2015) proposes to sample from the optimal posterior approximation instead. By applying Jensen's inequality to bound the log joint marginal we obtain the following formulation,

$$\log p(\mathbf{y}, \boldsymbol{\Psi} \mid \mathbf{X}) \geqslant \mathbb{E}_{p(\mathbf{f} \mid \boldsymbol{\Psi}, \mathbf{X})} \log p(\mathbf{y} \mid \mathbf{f}) + \log p(\boldsymbol{\Psi}) \overset{\text{def}}{=} \log \widetilde{p}_{\text{VFE}}(\mathbf{y}, \boldsymbol{\Psi} \mid \mathbf{X}). \tag{6.3}$$

This is the same expression derived in Hensman, A. G. Matthews, et al. (2015), although following a different derivation showing that $\widetilde{p}_{\text{VFE}}$ indeed yields the optimal distribution under the VFE framework of Equation (6.2). However, Hensman, A. G. Matthews, et al., 2015 argues that a Bayesian treatment of inducing inputs is unnecessary and concludes that the optimal prior is $p(\mathbf{Z}) = q(\mathbf{Z}) = \delta(\mathbf{Z} - \widehat{\mathbf{Z}})$, where $\delta(\cdot)$ is Dirac's delta function and $\widehat{\mathbf{Z}}$ is the set of inducing inputs that maximizes the ELBO (Hensman, A. G. Matthews, et al., 2015, §3). We find such a justification flawed as it contradicts the fundamental principles of Bayesian inference. Indeed, the derivation in (Hensman, A. G. Matthews, et al., 2015) relies on minimizing both sides of the KL term in Equation (6.2), allowing for a 'free-form' optimization of the prior, which ultimately negates the necessity of all prior choices and defeats the purpose of a Bayesian treatment.

FITC APPROXIMATIONS. As an alternative, we can approximate the log joint of Equation (6.1) by imposing independence in the conditional distribution (see Quiñonero-Candela and Rasmussen, 2005, for details),

$$p(\mathbf{f} \mid \boldsymbol{\Psi}, \mathbf{X}) = \mathcal{N}\left(\mathbf{K}_{\mathrm{xz}|\theta} \mathbf{K}_{\mathrm{zz}|\theta}^{-1} \mathbf{u}, \operatorname{diag}\left[\mathbf{K}_{\mathrm{xx}|\theta} - \mathbf{K}_{\mathrm{xz}|\theta} \mathbf{K}_{\mathrm{zz}|\theta}^{-1} \mathbf{K}_{\mathrm{xz}|\theta}^{\top}\right]\right) \tag{6.4}$$

which implies,

$$\log p(\mathbf{y}, \boldsymbol{\Psi} \mid \mathbf{X}) \approx \underbrace{\sum_{n=1}^{N} \log \mathbb{E}_{p(f_n \mid \boldsymbol{\Psi}, \mathbf{X})} \left[p(y_n \mid f_n)\right] + \log p(\boldsymbol{\Psi})}_{\overset{\text{def}}{=} \log \widetilde{p}_{\text{FITC}}(\mathbf{y}, \boldsymbol{\Psi} \mid \mathbf{X})}. \tag{6.5}$$

This same formulation of the FITC objective can be also been obtain by modifying the likelihood or the prior rather then the conditional distribution (E. Snelson and Ghahramani, 2006; Titsias, 2009b; Bauer et al., 2016).

We now see that, when considering i.i.d. conditional likelihoods, both approximations, $\log \widetilde{p}_{\text{VFE}}$ and $\log \widetilde{p}_{\text{FITC}}$, yield objectives that decompose on the observations, enabling scalable inference methods. In particular, we aim to sample from the posterior over all the latent variables using scalable approaches such as Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) (T. Chen et al., 2014). The main question is what approach should be preferred and how they relate to their optimization counterparts.

6.2.2 Sampling with VFE or FITC?

We will show in § 6.4 that our proposal that samples from the posterior according to Equation (6.5) consistently outperforms that in Equation (6.3). To understand why the FITC objective makes sense we need to go back to the original work of Titsias (2009b) and Titsias (2009a) and the seminal work of Quiñonero-Candela and Rasmussen (2005). For this, we will consider the regression case but it can be easily generalized to classification. Indeed, Titsias, 2009a shows that, in the standard regression case with homoskedastic observation noise, VFE yields exactly the same predictive posterior as the projected process (PP) approximation (Seeger et al., 2003), which is referred to as the deterministic training conditional (DTC) approximation. The optimal variational posterior distribution is given by:

$$\begin{aligned} q^*(\mathbf{u} \mid \theta) &= \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S}), \\ \mathbf{m} &= \sigma^{-2} \mathbf{K}_{zz} \left(\mathbf{K}_{zz} + \sigma^{-2} \mathbf{K}_{zx} \mathbf{K}_{xz} \right)^{-1} \mathbf{K}_{zx} \mathbf{y} \\ \mathbf{S} &= \mathbf{K}_{zz} \left(\mathbf{K}_{zz} + \sigma^{-2} \mathbf{K}_{zx} \mathbf{K}_{xz} \right)^{-1} \mathbf{K}_{zz} \end{aligned} \tag{6.6}$$

where σ^2 is the observation-noise variance. It is easy to show that, given a Gaussian posterior over the inducing variables with mean and covariance $\mathbf{m}$ and $\mathbf{S}$, the posterior predictive distribution at test point $\mathbf{x}_\star$ is a Gaussian with mean and variance

$$\begin{aligned} \mu_y(\mathbf{x}_\star) &= \kappa(\mathbf{x}_\star, \mathbf{Z}) \mathbf{K}_{zz}^{-1} \mathbf{m} \\ \sigma_y^2(\mathbf{x}_\star) &= \kappa(\mathbf{x}_\star, \mathbf{x}_\star) - \kappa(\mathbf{x}_\star, \mathbf{Z}) \mathbf{K}_{zz}^{-1} \kappa(\mathbf{Z}, \mathbf{x}_\star) + \kappa(\mathbf{x}_\star, \mathbf{Z}) \mathbf{K}_{zz}^{-1} \mathbf{S} \mathbf{K}_{zz}^{-1} \kappa(\mathbf{Z}, \mathbf{x}_\star). \end{aligned} \tag{6.7}$$

Thus, replacing Equation (6.6) in Equation (6.7) we obtain:

$$\begin{aligned} \mu_y(\mathbf{x}_\star) &= \sigma^{-2} \kappa(\mathbf{x}_\star, \mathbf{Z}) \boldsymbol{\Sigma} \mathbf{K}_{zx} \mathbf{y} \\ \sigma_y^2(\mathbf{x}_\star) &= \kappa(\mathbf{x}_\star, \mathbf{x}_\star) - \kappa(\mathbf{x}_\star, \mathbf{Z}) \mathbf{K}_{zz}^{-1} \kappa(\mathbf{Z}, \mathbf{x}_\star) + \kappa(\mathbf{x}_\star, \mathbf{Z}) \left(\mathbf{K}_{zz} + \sigma^{-2} \mathbf{K}_{zx} \mathbf{K}_{xz} \right)^{-1} \kappa(\mathbf{Z}, \mathbf{x}_\star), \end{aligned} \tag{6.8}$$

which indeed corresponds to the predictive distribution of the DTC/PP approximation. Despite this equivalence, as highlighted in Titsias, 2009b, the main difference is that the VFE framework provides a more robust approach to hyper-parameter estimation as the resulting ELBO corresponds to a regularized marginal likelihood of the DTC approach and hence should be more robust to overfitting. Nevertheless, the DTC/PP, and consequently the VFE, predictive distribution has been shown to be less accurate than the FITC approximation (Titsias, 2009b; Quiñonero-Candela and Rasmussen, 2005; E. L. Snelson, 2007). Effectively, as described in Quiñonero-Candela and Rasmussen, 2005, the VFE's solution (which is the same as DTC's) can be understood as considering a deterministic conditional prior $p(\mathbf{f}\,|\,\mathbf{u})$, i.e. a conditional prior with zero variance.

The FITC approximation considers the following approximate conditional prior:

$$\begin{aligned} p(\mathbf{f}\,|\,\mathbf{u}) &\approx \mathcal{N}\left(\mathbf{f}; \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{u}, \operatorname{diag}\left(\mathbf{K}_{xx} - \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{K}_{zx}\right)\right) \\ &= \prod_{n=1}^{N} p(f_n\,|\,\mathbf{u}) = \prod_{n=1}^{N} \mathcal{N}(f_n; \tilde{\mu}_n, \tilde{\sigma}_n^2), \text{ with} \\ \tilde{\mu}_n &= \kappa(\mathbf{x}_n, \mathbf{Z})\mathbf{K}_{zz}^{-1}\mathbf{u} \qquad (6.9) \\ \tilde{\sigma}_n^2 &= \kappa(\mathbf{x}_n, \mathbf{x}_n) - \kappa(\mathbf{x}_n, \mathbf{Z})\mathbf{K}_{zz}^{-1}\kappa(\mathbf{Z}, \mathbf{x}_n). \qquad (6.10) \end{aligned}$$

As we shall see later, is this factorization assumption in the conditional prior that will yield a decomposable objective amenable to stochastic gradient techniques. For now, consider the posterior predictive distribution under the FITC approximation[1]

$$\begin{aligned} \mu_{\text{FITC}}(\mathbf{x}_\star) &= \kappa(\mathbf{x}_\star, \mathbf{Z})\boldsymbol{\Sigma}_{\text{FITC}}\mathbf{K}_{zx}\boldsymbol{\Lambda}^{-1}\mathbf{y} \\ \sigma^2_{\text{FITC}}(\mathbf{x}_\star) &= \kappa(\mathbf{x}_\star, \mathbf{x}_\star) - \kappa(\mathbf{x}_\star, \mathbf{Z})\mathbf{K}_{zz}^{-1}\kappa(\mathbf{Z}, \mathbf{x}_\star) + \kappa(\mathbf{x}_\star, \mathbf{Z})\boldsymbol{\Sigma}_{\text{FITC}}\kappa(\mathbf{Z}, \mathbf{x}_\star), \text{ where} \\ \boldsymbol{\Lambda} &= \operatorname{diag}(\mathbf{K}_{xx} - \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{K}_{zx} + \sigma^2\mathbf{I}) \text{ and} \\ \boldsymbol{\Sigma}_{\text{FITC}} &= (\mathbf{K}_{zz} + \mathbf{K}_{zx}\boldsymbol{\Lambda}^{-1}\mathbf{K}_{xz})^{-1}. \end{aligned} \qquad (6.11)$$

We now see why FITC's predictive distribution above is more accurate than VFE's in Equation (6.8), as we can obtain FITC's by replacing $\sigma^2\mathbf{I}$ in VFE's solution with $\operatorname{diag}(\mathbf{K}_{xx} - \mathbf{K}_{xz}\mathbf{K}_{zz}^{-1}\mathbf{K}_{zx}) + \sigma^2\mathbf{I}$. Effectively, as described in Quiñonero-Candela and Rasmussen, 2005, VFE's solution (which is the same as DTC's) can be understood as considering a deterministic conditional prior $p(\mathbf{f}\,|\,\mathbf{u})$, i.e. with zero variance.

6.2.3 Stochastic Updates Using the FITC Approximation

Now we can understand why the log of the expectation can provide more accurate results than the expectation of the log. Basically in the former we are using the FITC approximation while in the later we are using the VFE/DTC/PP approximation. It is easy to show that when using the FITC approximation, one can obtain a decomposable objective function that can be implemented at large scale using stochastic gradient techniques. Here we focus only on the expectation of the conditional likelihood (which is the crucial term) and in the regression setting for simplicity but the extension to the classification case (e.g. using quadrature) is straightforward.

---

[1] Which is, in fact, the same as in the sparse Gaussian process (SPGP) framework of E. L. Snelson (2007).

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{u}, \theta, \mathbf{Z}) &= \log \int_{\mathbf{f}} p(\mathbf{f} \mid \mathbf{u}, \theta, \mathbf{Z}) p(\mathbf{y} \mid \mathbf{f}) \, d\mathbf{f} \\
&= \log \int_{f_1, \dots, f_N} \prod_{n=1}^{N} p(y_n \mid f_n) p(f_n \mid \mathbf{u}, \theta, \mathbf{Z}) \, d\mathbf{f} \\
&= \log \prod_{n=1}^{N} \int_{f_n} \mathcal{N}(y_n \mid f_n, \sigma^2) \mathcal{N}(f_n \mid \tilde{\mu}_n, \tilde{\sigma}_n^2) \, df_n \\
&= \log \prod_{n=1}^{N} p(y_n \mid \mathbf{u}, \theta, \mathbf{Z}) \\
&= \sum_{n=1}^{N} \log \mathcal{N}(y_n \mid \tilde{\mu}_n, \tilde{\sigma}_n^2 + \sigma^2),
\end{aligned} \tag{6.12}$$

where $\tilde{\mu}_n, \tilde{\sigma}_n^2$ are given by Equation (6.9) and Equation (6.10). Similar results can be derived for binary classification with Bernoulli likelihood and response function $\lambda(f)$:

$$\log p(\mathbf{y} \mid \mathbf{u}\theta) = \log \mathbb{E}_{p(\mathbf{f} \mid \mathbf{u}, \theta)} [p(\mathbf{y} \mid \mathbf{f})] = \log \prod_{n=1}^{N} \int_{f_n} \mathcal{N}(f_n; \tilde{\mu}_n, \tilde{\sigma}_n^2) \mathrm{Bern}(y_n; \lambda(f_n)) \, df_n. \tag{6.13}$$

When the response function is the cdf of a standard Normal distribution, i.e., $\lambda(f_n) = \Phi(f_n) \overset{\text{def}}{=} \int_{-\infty}^{f_n} \mathcal{N}(f_n; 0, 1) df_n$, which is also known as the probit regression model, the expectation above can be computed analytically to obtain:

$$\log p(\mathbf{y}, \mathbf{u} \mid \theta) = \sum_{n=1}^{N} \log \mathrm{Bern}(y_n; \Phi(\tilde{\mu}_n / \sqrt{1 + \tilde{\sigma}_n^2})). \tag{6.14}$$

For other response functions the expectation in Equation (6.13) can be estimated using quadrature.

6.2.4 An heteroskedastic version of the Gaussian likelihood

As Titsias (2009a) discussed in Appendix C, the FITC approximation corresponds to a GP regression with heteroskedastic noise variance

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I} + \operatorname{diag}[\mathbf{K}_{\mathbf{x}\mathbf{x}} - \mathbf{K}_{\mathbf{x}\mathbf{z}}\mathbf{K}_{\mathbf{z}\mathbf{z}}\mathbf{K}_{\mathbf{z}\mathbf{x}}]). \tag{6.15}$$

If we apply this augmented likelihood to the variational expectations term, we get

$$\mathbb{E}_{q(\mathbf{f})} \log p(\mathbf{y}|\mathbf{f}, \sigma^2, \boldsymbol{\theta}) = -\frac{1}{2}\sum_{j=1}^{n}\left(\log 2\pi(\sigma^2 + \widetilde{\sigma}_j^2) + \frac{(y_j - \widetilde{\mu}_j)^2 + \widetilde{\sigma}_j^2}{\sigma^2 + \widetilde{\sigma}_j^2}\right). \tag{6.16}$$

Since Titsias, 2009a considers this VFE formulation, we also compare with it.

6.2.5 Concluding Remarks

We conclude this section by placing our analysis in the context of 'exact' Bayesian inference techniques that consider priors over the inducing inputs. As mentioned above, the main reason for the superior performance of VFE, despite providing a less accurate predictive posterior than FITC's, was that inducing input estimation was more robust due to the use of the variational objective, which provided an extra regularization term. However, by placing priors over the inducing inputs as well as over covariance hyper-parameters, the problem of regularization over these parameters becomes irrelevant. It is important to highlight that a variational formulation equivalent to FITC has also been proposed (see Titsias, 2009a, App. C). In our experiments in § 6.4 we show that, nonetheless, our fully Bayesian formulation still yields superior performance to such an approach, confirming the benefits of carrying out full posterior estimation over the inducing inputs.

A final remark is that, in the spirit of Bayesian modeling, any uncertainty in the covariance should be accounted for. Thinking of GP hyper-parameters and inducing inputs as parameters of the covariance function, a distribution over these induces a distribution over the covariance function, which enriches the modeling capabilities of these models (see, e.g., (Jang et al., 2017) or Figure 6.1).

6.3 PRACTICAL CONSIDERATIONS AND EXTENSIONS TO DEEP GPS

In this section we describe practical considerations in our Bayesian Sparse Gaussian Process (BSGP) framework, including inference techniques, prior choices and extensions to deep Gaussian processes. Recalling that $\boldsymbol{\Psi} = \{\theta, \mathbf{u}, \mathbf{Z}\}$ represents the set of variables to infer and, using Equation (6.5), their posterior can be obtained as

$$\begin{aligned}\log p(\boldsymbol{\Psi} \mid \mathbf{y}, \mathbf{X}) = \log \mathbb{E}_{p(\mathbf{f} \mid \boldsymbol{\Psi}, \mathbf{X})} p(\mathbf{y} \mid \mathbf{f}) + \log p(\mathbf{u} \mid \theta, \mathbf{Z}) + \\ \log p_{\xi}(\mathbf{Z}) + \log p_{\psi}(\theta) - \log C. \end{aligned} \tag{6.17}$$

We use Markov chain Monte Carlo (MCMC) techniques, in particular SGHMC (T. Chen et al., 2014; Havasi et al., 2018), to obtain samples from the intractable $p(\boldsymbol{\Psi} \mid \mathbf{y}, \mathbf{X})$. Unlike Hamiltonian Monte Carlo (HMC), which requires computing the exact gradient $\nabla \log p(\boldsymbol{\Psi} \mid \mathbf{y}, \mathbf{X})$ and the exact unnormalized posterior to evaluate the acceptance (Neal, 2011), SGHMC obtains samples from the posterior with stochastic gradients and without evaluating the Metropolis ratio (see supplement for details). With a factorized likelihood $p(\mathbf{y}|\mathbf{f})$ and an energy function $U(\boldsymbol{\Psi}) = -\log p(\boldsymbol{\Psi} \mid \mathbf{y}, \mathbf{X}) + \log C$, we sample Equation (6.17) over minibatches of data.

6.3.1 Prior choices

Next, we discuss prior choices for the inducing inputs and covariance hyperparameters. The inducing inputs $\mathbf{Z}$ support the sparse Gaussian process interpolation, which motivates matching the inducing prior to the data distribution $p(\mathbf{X})$. We begin by proposing a simple Normal (N) prior

$$p_N(\mathbf{Z}) = \prod_{j=1}^{M} \mathcal{N}(\mathbf{z}_j | \mathbf{0}, \mathbf{I}), \tag{6.18}$$

which matches the mean and variance of the normalized data distribution, and favors inducing inputs toward the baricenter of the data inputs.

We also explore two priors based on point processes, which consider distributions over point sets (Gonzalez et al., 2016). Point processes can induce repulsive ef-

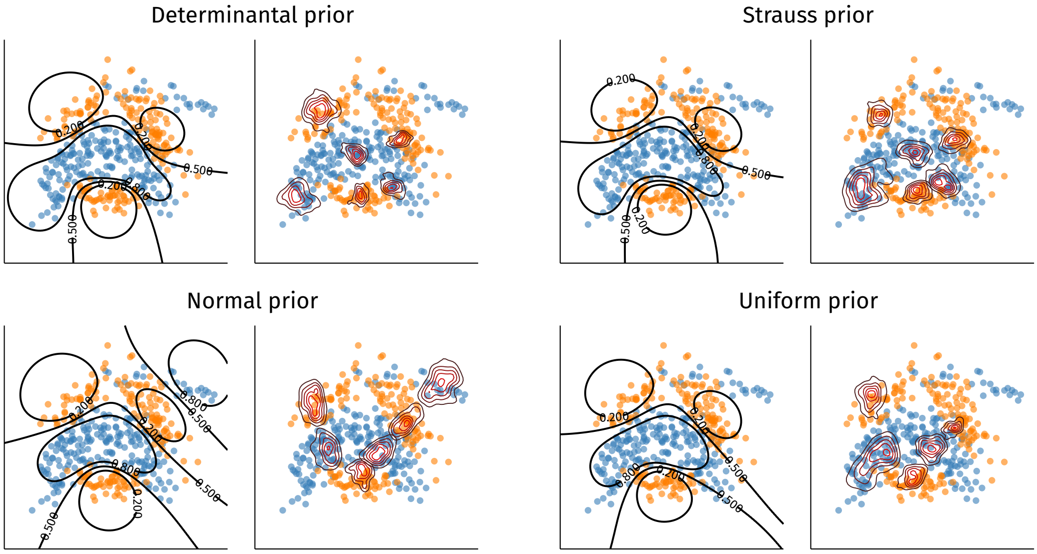


Figure 6.2: Illustration of a binary classification task on the BANANA dataset. *Left*: the decision bounds of the average classifier. *Right*: the posterior marginals of the inducing inputs.

fects penalizing configurations where inducing points are clumped together. The determinantal point process (DPP), defined through

$$p_{D}(\mathbf{Z}) \propto \det \mathbf{K}_{\mathbf{zz}|\theta} \,, \tag{6.19}$$

relates the probability of inducing inputs to the volume of space spanned by the covariance (Lavancier et al., 2015). DPP is a repulsive point process, which gives higher probabilities to input diversity, controlled by the hyper-parameters $\xi \equiv \theta$. We then consider the Strauss process (see e.g. Daley and Vere-Jones, 2003; Strauss, 1975),

$$p_{S}(\mathbf{Z}) \propto \lambda^{M} \gamma^{\sum_{z,z' \in Z} \delta(|z-z'|<r)}, \tag{6.20}$$

where $\lambda > 0$ is the intensity, and $0 < \gamma \leqslant 1$ is the repulsion coefficient which decays the prior as a function of the number of input pairs that are within distance r. The Strauss prior (S) tends to maintain the minimum distance between inducing inputs, parameterized by $\xi = (\lambda, \gamma, r)$. We finally consider an uninformative uniform prior (U), $\log p_{U}(\mathbf{Z}) = 0$, which effectively provides no contribution to the evaluation of the posterior.

To gain insights on the choice of these priors, we set up a comparative analysis on the BANANA dataset (Figure 6.2). We observe that the posterior densities on the inducing inputs are multimodal and highly non-Gaussian, further confirming

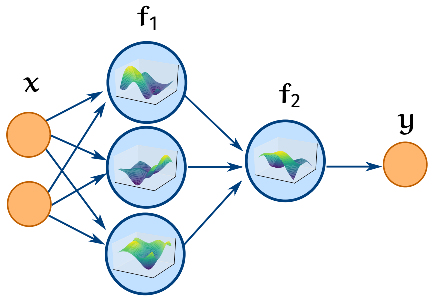


Figure 6.3: Visual representation of a 2-layer deep Gaussian process (DGP).

the necessity of free-form inference. Both Strauss and DPP-based priors encourage configurations where the inducing inputs are evenly spread. The Normal and Uniform priors, instead, focus exclusively on aligning the inducing inputs in a way that is sensible to accurately model the intricate classification boundary between the classes. This insight is confirmed by our the extensive experimental validation in § 6.4.

PRIOR ON COVARIANCE HYPER-PARAMETERS. Choosing priors on the hyper-parameters has been discussed in previous works on Bayesian inference for GPs (see e.g. Filippone and Girolami, 2014). Throughout this chapter, we use the RBF covariance with automatic relevance determination (ARD), marginal variance σ and independent lengthscales λ_i per feature (Mackay, 1994). On these two hyper-parameters we place a lognormal prior with unit variance and means equal to 1 and 0.05 for $\boldsymbol{\lambda}$ and σ, respectively.

6.3.2 Extension to deep Gaussian processes

BSGP can be easily extended to DGP models (Damianou and Lawrence, 2013). Deep Gaussian processes are hierarchical compositions of L sparse GPs generally defined as

$$\mathcal{DGP}(\mathbf{x}) = (\mathbf{f}_L \circ \ldots \circ \mathbf{f}_1)(\mathbf{x})$$

Each layer $\mathbf{f}_l$ is a vector-valued GP:

$$\mathbf{f}_i(\cdot) = \left[f_i^{(1)}(\cdot), \ldots, f_i^{(H_i)}(\cdot)\right]^\top \qquad \text{with} \qquad f_i^{(j)}(\cdot) \overset{\text{i.i.d}}{\sim} \mathcal{GP}(0, \kappa_i(\cdot, \cdot \mid \boldsymbol{\theta}_i)).$$

where H_i is the width of the i^{th} GP which are taken to be independent. Each layer is associated with a set of inducing inputs $\mathbf{Z}^{(l)}$, inducing variables $\mathbf{u}^{(l)}$

and hyper-parameters $\theta^{(l)}$ (Salimbeni and Deisenroth, 2017). In our notation $\mathbf{\Psi} = \left\{\mathbf{\Psi}^{(l)}\right\}_{\ell=1}^{L} = \left\{\mathbf{Z}^{(l)}, \mathbf{u}^{(l)}, \theta^{(l)}\right\}_{l=1}^{L}$.

In this section, we derive the mathematical basis for a Bayesian treatment of inducing inputs in a DGP setting (Damianou and Lawrence, 2013). We assume a deep Gaussian process prior $f^{(L)} \circ f^{(L-1)} \circ \cdots f^{(1)}$, where each $f^{(l)}$ is a GP. For notational brevity, we use $\mathbf{f}^{(0)}$ as the input vector $\mathbf{x}$. Then we can write down the joint distribution over visible and latent variables (omitting the dependency on $\mathbf{X}$ for clarity) as

$$p\left(\mathbf{y}, \left\{\mathbf{f}^{(l)}, \mathbf{\Psi}^{(l)}\right\}_{l=1}^{L}\right) = p\left(\mathbf{y} \mid \mathbf{f}^{(L)}\right) \prod_{l=1}^{L} p\left(\mathbf{f}^{(l)} \mid \mathbf{\Psi}^{(l)}, \mathbf{f}^{(l-1)}\right) p\left(\mathbf{\Psi}^{(l)}\right). \tag{6.21}$$

Our goal is to estimate the posterior after marginalizing out $\mathbf{f}^{(l)}$,

$$\begin{aligned}
&\log p\left(\left\{\psi^{(l)}\right\}_{l=1}^{L} \mid \mathbf{y}\right) \propto \\
&= \log \int p\left(\mathbf{y} \mid \mathbf{f}^{(L)}\right) p\left(\mathbf{f}^{(L)} \mid \psi^{(L)}, \mathbf{f}^{(L-1)}\right) \ldots p\left(\mathbf{f}^{(1)} \mid \psi^{(1), \mathbf{f}^{(0)}}\right) d\mathbf{f}^{(L)} d\mathbf{f}^{(L-1)} \ldots d\mathbf{f}^{(1)} \\
&\quad + \sum_{l=1}^{L} \log p(\mathbf{\Psi}^{(l)}).
\end{aligned} \tag{6.22}$$

While the distribution in Equation (6.22) is not immediately computable owing to the intractable expectation term, we have obtained the form of its (un-normalized) log posterior, from which we can sample using HMC methods. More calculations reveal that we can, nevertheless, obtain estimates of this expectation term with Monte Carlo sampling

$$\begin{aligned}
&\log \int p\left(\mathbf{y} \mid \mathbf{f}^{(L)}\right) p\left(\mathbf{f}^{(L)} \mid \psi^{(L)}, \mathbf{f}^{(L-1)}\right) \ldots p\left(\mathbf{f}^{(1)} \mid \psi^{(1)}\right) d\mathbf{f}^{(L)} d\mathbf{f}^{(L-1)} \ldots d\mathbf{f}^{(1)} \approx \\
&\log \int p\left(\mathbf{y} \mid \mathbf{f}^{(L)}\right) p\left(\mathbf{f}^{(L)} \mid \psi^{(L)}, \widetilde{\mathbf{f}^{(L-1)}}\right) d\mathbf{f}^{(L)}
\end{aligned} \tag{6.23}$$

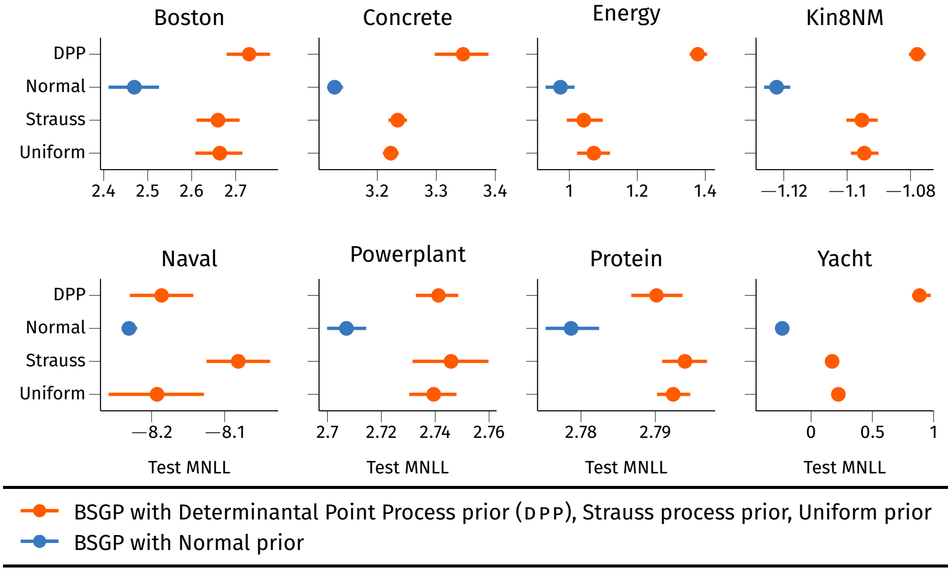


Figure 6.4: Analysis of different priors on inducing locations for BSGP on the UCI benchmark datasets for determinantal point process (DPP), Strauss process, uniform and normal priors on $\mathbf{Z}$.

where,

$$
\begin{aligned}
\widetilde{\mathbf{f}^{(1)}} &\sim p\left(\mathbf{f}^{(1)} \mid \psi^{(1)}, \mathbf{f}^{(0)}\right) \\
\widetilde{\mathbf{f}^{(2)}} &\sim p\left(\mathbf{f}^{(2)} \mid \psi^{(2)}, \widetilde{\mathbf{f}^{(1)}}\right) \\
&\cdots \\
\widetilde{\mathbf{f}^{(L-1)}} &\sim p\left(\mathbf{f}^{(L-1)} \mid \psi^{(L-1)}, \widetilde{\mathbf{f}^{(L-2)}}\right)
\end{aligned}
$$

Because of the layer-wise factorization of the joint likelihood (Equation (6.21)), each step of the approximation is unbiased. While it is possible to approximate the last-layer expectation with a Monte Carlo sample, the expectation is tractable when the likelihood is a Gaussian or a Bernoulli distribution with a probit regression model, or is computable with one-dimensional quadrature (Hensman, A. G. Matthews, et al., 2015).

6.4 EXPERIMENTS

In this section, we provide empirical evidence that our BSGP outperforms previous inference/optimization approaches on shallow and deep GPs. We use eight of the

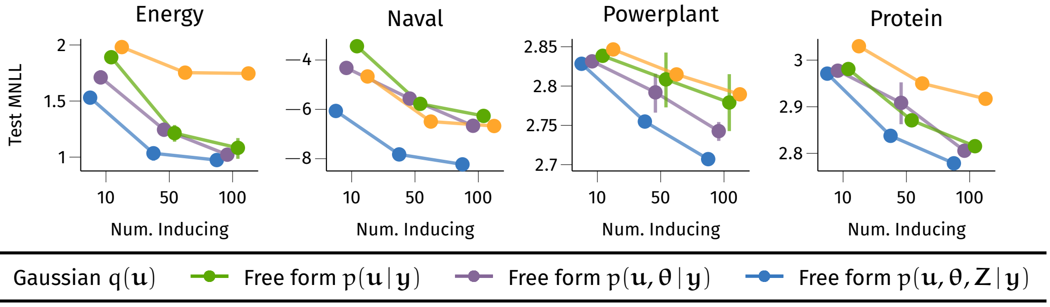


Figure 6.5: Ablation study on the effect of performing posterior inference on different sets of variables. From SVGP, where the posterior is constrained to be Gaussian and the remaining parameters are point-estimated, to our proposal BSGP, where we infer a free-form posterior for all $\Psi = \{u, \theta, Z\}$. We refer the reader to Table 6.1 for details on the methods (colors are matched).

classic UCI benchmark datasets with standardized features and split into eight folds with 0.8/0.2 train/test ratio. We train the competing models for 10,000 iterations with ADAM (Kingma and Ba, 2015), step size of 0.01 and a minibatch of 1,000 samples. The sampling methods are evaluated based on 256 samples collected after optimization. Following previous works (e.g. Rasmussen and Williams, 2005; Havasi et al., 2018; H. Yu et al., 2019), in order to evaluate and compare the full predictive posteriors we compute the mean negative loglikelihood (MNLL) on the test set (RMSEs are reported in the supplement for reference).

6.4.1 Prior analysis and ablation study

We start our empirical analysis with a comparative evaluation of the priors on inducing inputs described in § 6.3.1: DPP, Normal, Strauss and Uniform. We run our inference procedure on a shallow GP with 100 inducing points and we report the results in **??** (left). The results show that the Normal prior consistently outperforms the others. The uniform and Strauss priors behave similarly, while the DPP prior is consistently among the worst. We argue that the repulsive nature of the point process priors (DPP, particularly), although grounded on the intuition of covering the input space more evenly, constrains the smoothness of the functions up to the point that they become too simple to accurately model the data. With this, we select the Gaussian prior for the remaining experiments.

We now study the benefits of a Bayesian treatment of the inducing variables, inducing inputs, and hyper-parameters with an ablation study. Using the same setup as before, we start with the baseline of SVGP (Hensman, Fusi, et al., 2013;

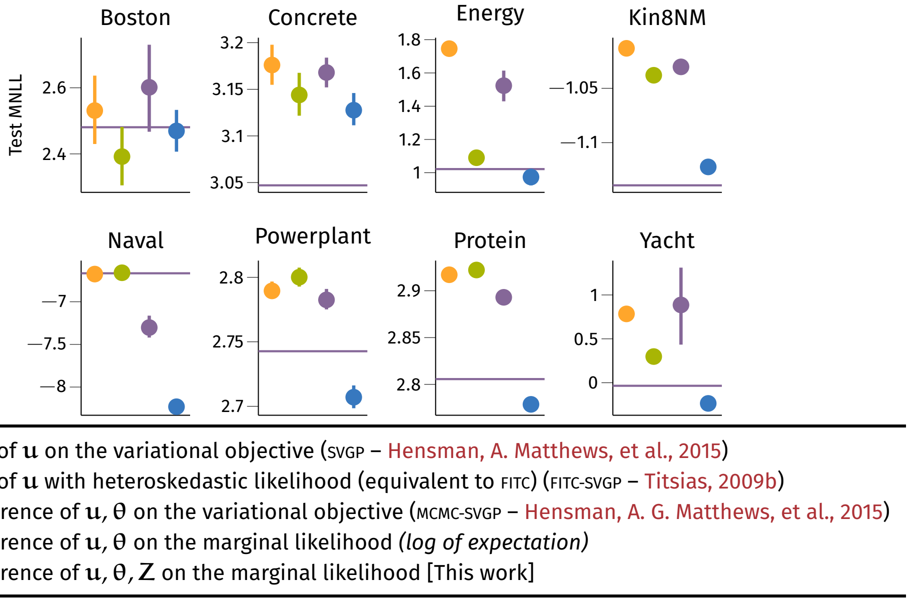


Figure 6.6: Analysis of different choices of objectives when used for optimization and sampling. We refer the reader to Table 6.1 for a description of the methods.

Hensman, A. Matthews, et al., 2015), where the posterior on $\mathbf{u}$ is approximated using a Gaussian and $\mathbf{Z}, \theta$ are optimized. We then incrementally add parameters to the list of variables that are sampled rather than optimized: only $\mathbf{u}$ (equivalent to SGHMC-DGP, (Havasi et al., 2018)), then $\{\mathbf{u}, \theta\}$ and finally, our proposal, $\{\mathbf{u}, \theta, \mathbf{Z}\}$. This experiment is repeated for different number of inducing points (10, 50 and 100). **??** (right) reports a summary of these results (full comparison in the supplement). This plot shows that each time we carry out free-form posterior inference on a bigger set of parameters rather than optimization, performance is enhanced, and our proposal outperforms previous approaches.

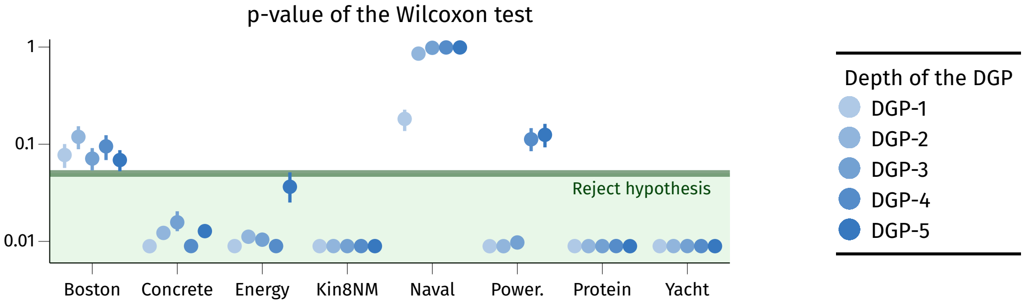


Figure 6.7: p-values of the hypothesis test that BSGP with VFE objective is better than BSGP with FITC objective; depth of the DGP from 1 to 5. For models with p-values < 0.05, we reject the hypothesis

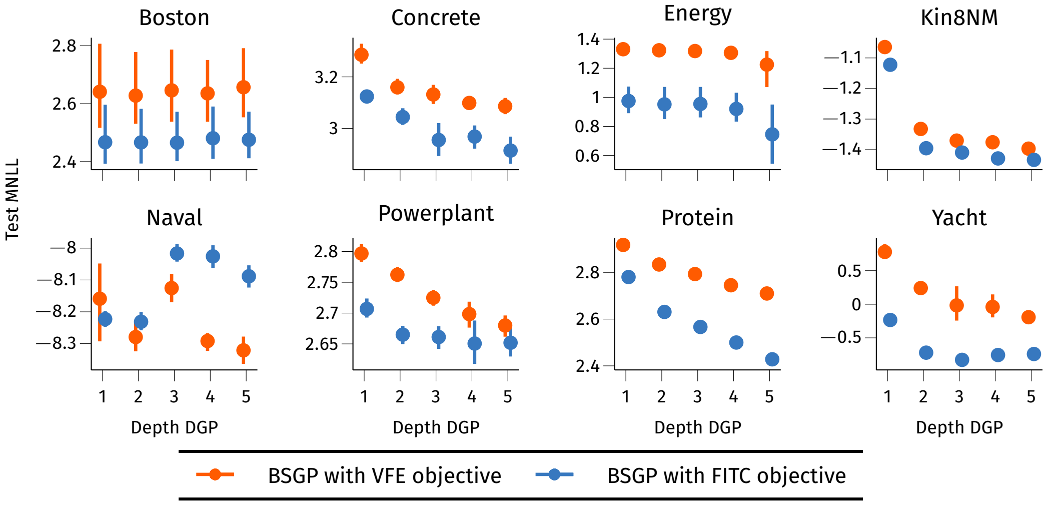


Figure 6.8: BSGP in with different depths of the DGP with two different objective: FITC and VFE. The number of layers corresponds to the depth of the DGP.

6.4.2 Choosing the objective: VFE vs FITC

In § 6.2 we discussed the role of the marginal and the variational free energy (VFE) objective when used for optimization and for sampling. In Figure 6.6 we support the discussion with empirical results. The baseline is SVGP, for which the inference is approximate (Gaussian) and performed on the variational objective. Titsias (2009a, App. C) also considers a VFE formulation of FITC which corresponds to a GP regression with heteroskedastic noise variance. The likelihood needs to be augmented to handle heteroskedasticity, but inference can be carried out exactly on the variational objective. For these two methods, $\{\theta, \mathbf{Z}\}$ are optimized. We also test MCMC-SVGP, the model proposed by Hensman, A. G. Matthews, et al. (2015), implemented in GPflow (A. G. Matthews et al., 2017) with the same suggested experimental setup. This experiment indicates that having a free-form posterior on $\mathbf{u}, \theta$ sampled from the variational objective does not dramatically improve on the exact Gaussian approximation of the FITC model, with both of them delivering superior performance with respect to SVGP. In the same setup of Hensman, A. G. Matthews, et al. (2015) ($\mathbf{u}, \theta$ sampled and $\mathbf{Z}$ optimized), we look at the effect of swapping the expectation of log with the log of expectation (which effectively means moving from the VFE objective to FITC); on the contrary, here we observe a significant increase in performance when using the latter, further confirming the discussion of the objectives in § 6.2.2. We finally conclude this section with an experiment where we try both objectives on our proposed BSGP and also different depths of the DGP (Figure 6.8). Using the Wilcoxon signed-rank test (Wilcoxon,

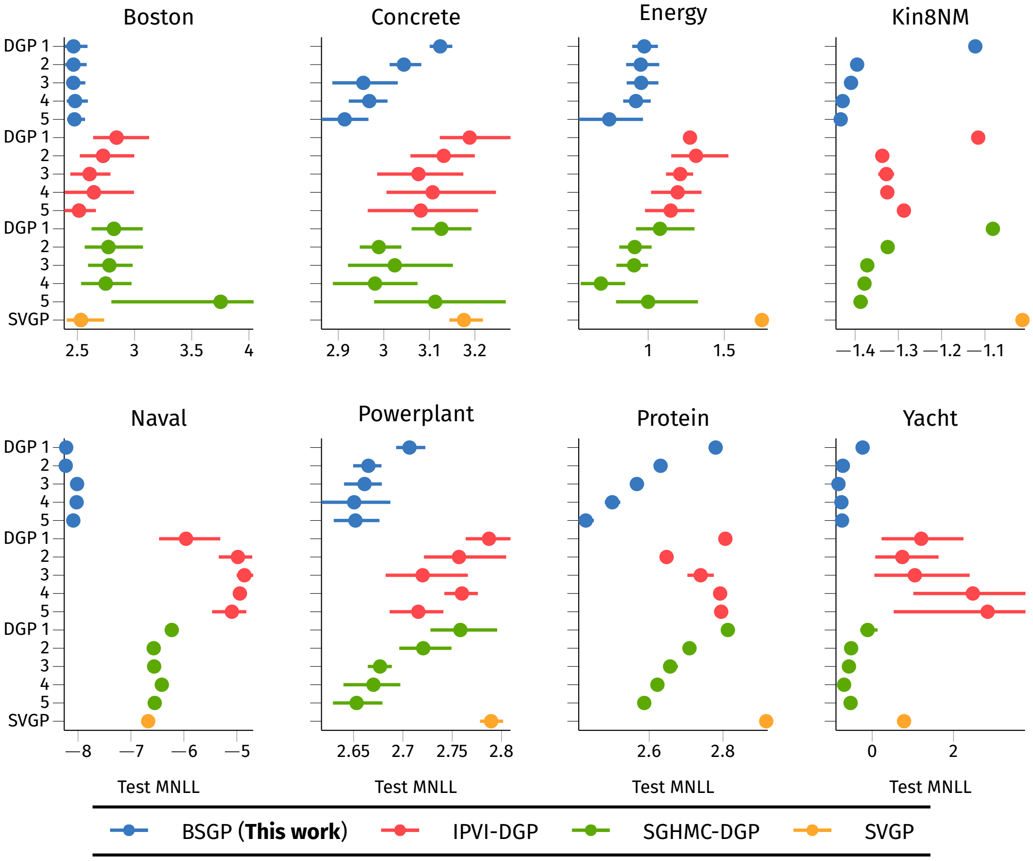


Figure 6.9: Test MNLL on UCI regression benchmarks (the error bars represent the 95%CI). The lower MNLL (i.e. to the left), the better. The number on the right of the method's name refers to the depth of the DGP. *Bottom right*: Rank summary of all methods.

1945), we test the null hypothesis of VFE objective being better than the proposed FITC. Figure 6.7 shows that, for the majority of the cases, this can be rejected ($p < 0.05$).

6.4.3 Deep Gaussian processes on UCI benchmarks

We now report results on DGPs. We compare against two current state-of-the-art deep GP methods, SGHMC-DGP (Havasi et al., 2018) and IPVI-DGP (H. Yu et al., 2019), and against the shallow SVGP baseline (Hensman, A. Matthews, et al., 2015). For a faithful comparison with IPVI-DGP we follow the recommended parameter configurations[2]. Using a standard setup, all models share $M = 100$ inducing points, the same RBF covariance with ARD and, for DGP, the same hidden dimensions (equal to the input dimension D). Figure 6.9 shows the predictive test MNLL mean

---

[2] We use the IPVI-DGP implementation available at `https://github.com/HeroKillerEver/ipvi-dgp`

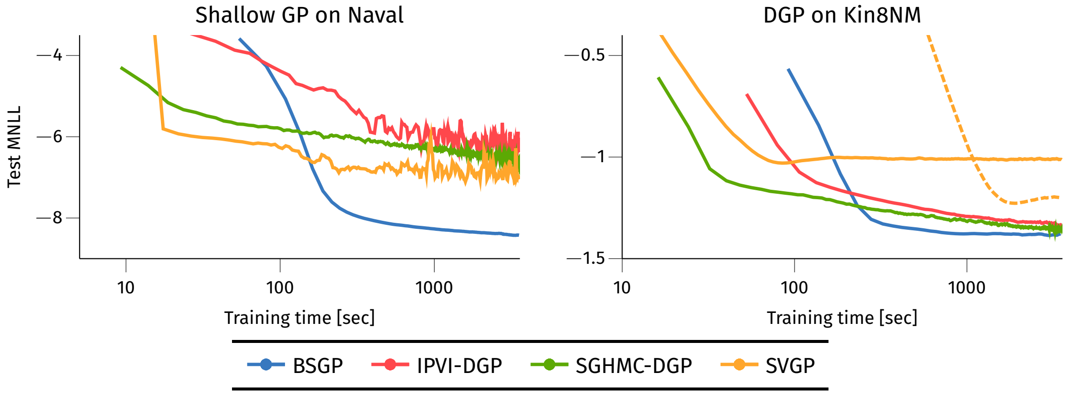


Figure 6.10: Comparison of test MNLL as function of training time. The dashed line on the right hand side plot corresponds to SVGP with $M = 1000$ inducing points.

and 95% CI over the different folds over the UCI datasets, and also includes rank summaries. The proposed method clearly outperforms competing deep and shallow GPs. The improvements are particularly evident on NAVAL, a dataset known to be challenging to improve upon. Furthermore, the deeper models perform consistently better or on par with the shallow version, without incurring in any measurable overfitting even on small or medium sized datasets (see BOSTON and YACHT, for example).

COMPUTATIONAL EFFICIENCY. Similarly to the baseline algorithms, each training iteration of BSGP involves the computation of the inverse covariance with complexity $\mathcal{O}(M^3)$. In Figure 6.10 we compare the three main competitors with BSGP trained for a fixed training time budget of one hour for a shallow GP and a 2-layer DGP. The experiment is repeated four times on the same fold and the results are then averaged. Each run is performed on an isolated instance in a cloud computing platform with 8 CPU cores and 8 GB of reserved memory (Pace et al., 2017). Inference on the test set is performed every 250 iterations. This shows that BSGP converges considerably faster in wall-clock time, even though a single gradient step requires slightly more time.

Computing the predictive distribution, on the other hand, is more challenging as it requires recomputing the covariance matrices $\mathbf{K}_{XZ}, \mathbf{K}_{ZZ}$ for each posterior sample $\{Z, \theta\}$, for an overall complexity linear in the number of posterior samples. This operation can be easily parallelized and implemented on GPUs but it could question the practicality of using a more involved inference method. In particular, it is relevant to study whether SVGP could deliver superior performance with a higher number of inducing points for less computational overhead. In Figure 6.11

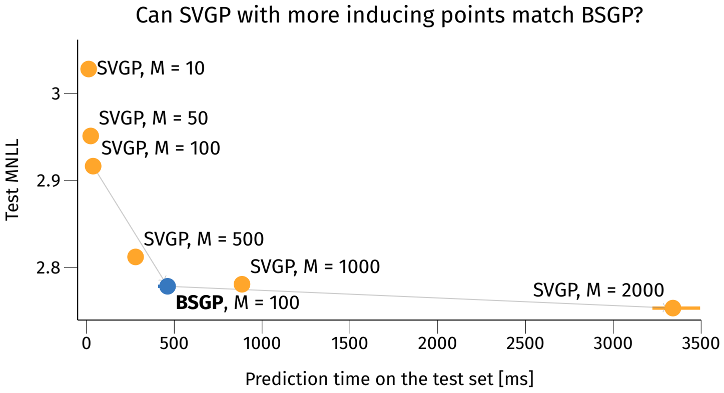


Figure 6.11: Comparison of test MNLL as a function of prediction time on the largest dataset (Protein).

we study this trade-off on the biggest dataset considered (Protein): while it is evident that predictions with BSGP take more time (assuming a serial computation of the covariance matrices), it is also clear that the number of inducing points SVGP requires to (even marginally) improve upon BSGP is significantly larger (up to 20 times).

STRUCTURED INDUCING POINTS. Finally, we run one last comparison with methods which exploit structure in the inputs. These models allow one to scale the number of inducing variables while maintaining computational tractability. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP) (A. Wilson and Nickisch, 2015) proposes to place the inducing inputs on a fixed and equally-spaced grid and to exploit Toeplitz/Kronecker structures with an iterative conjugate gradient method to further enhance scalability. Despite these benefits, KISS-GP is known to fall short with high-dimensional data ($D > 4$). This shortcoming was later addressed with Deep Kernel Learning (DKL) (A. G. Wilson, Hu, R. Salakhutdinov, et al., 2016): using a deep neural network DKL projects the data in a lower dimensional manifold by learning an useful feature representation, which is then used as input to a KISS-GP. In Figure 6.12 we have the comparison of BSGP with these two methods. KISS-GP could only run on Powerplant, with a 4-dimensional grid of size 10 (for a total of 10,000 inducing points). Here, BSGP delivers better performance despite having less inducing points. For DKL we followed the suggestion of A. G. Wilson, Hu, R. Salakhutdinov, et al. (2016) to use a fully-connected neural network with a $[d - 1000 - 1000 - 500 - 50 - 2]$ architecture as feature extractor and a grid size of 100 (for again a total of 10,000 inducing points). Training is performed by alternating optimization of the neural network weights and the KISS-GP parameters. Thanks

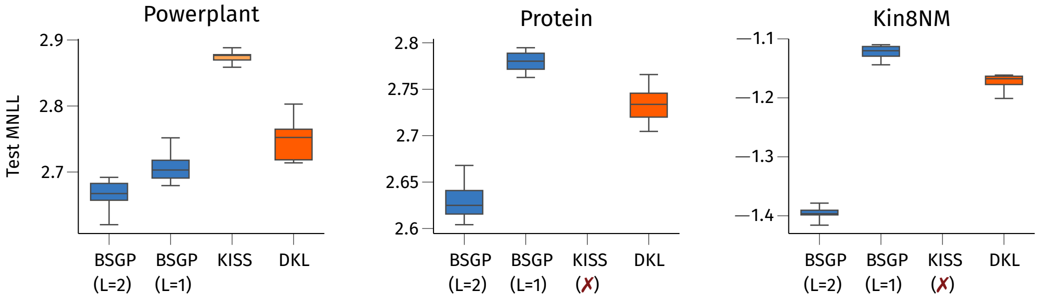


Figure 6.12: Comparison with structured inducing variables methods. KISS-GP could only run on the Powerplant dataset (hence the ✗ on Protein and Kin8NM).

to the flexibility of the feature extractor, this configuration is very competitive with our shallow BSGP, but it yields lower performance compared to a 2-layer DGP.

6.4.4 Large scale classification

The AIRLINE dataset is a classic benchmark for large scale classification. It collects delay information of all commercial flights in USA during 2008, counting more than 5 millions data points. The goal is to predict if a flight will be delayed based on 8 features, namely month, day of month, day of week, airtime, distance, arrival time, departure time and age of the plane. We pre-process the dataset following the guidelines provided in (Hensman, A. G. Matthews, et al., 2015; A. G. Wilson, Hu, R. Salakhutdinov, et al., 2016).

After a burn-in phase of 10,000 iterations, we draw 200 samples with 1000 simulation steps in between. We test on 100,000 randomly selected held-out points. We fit three models with $M = 100$ inducing points. Table 6.2 shows the predictive performance of three shallow GP models. The BSGP yields the best test error, MNLL, and test area under the curve (AUC). We assess the convergence of the predictive posterior by evaluating the $\hat{R}$-statistics (Gelman et al., 2004) over four independent SGHMC chains. This diagnostic yielded a $\hat{R} = 1.02 \pm 0.045$, which indicates good convergence. We report further convergence analysis in the supplement.

As a further large scale example, we use the HIGGS dataset (Baldi, Sadowski, et al., 2014), which has 11 millions data points with 28 features. This dataset was created by Monte Carlo simulations of particle dynamics in accelerators to detect the Higgs boson. We select 90% of the these points for training, while the

Table 6.2: AIRLINE dataset predictive test performance.

Model	Error (↓)	MNLL (↓)	AUC (↑)
SGHMC-GP	35.85%	0.646	0.671
SVGP	31.26%	0.595	0.730
BSGP	**30.46%**	**0.580**	**0.749**

Table 6.3: HIGGS dataset predictive test performance.

Model	Error (↓)	MNLL (↓)	AUC (↑)
SGHMC-GP	35.39%	0.628	0.698
SVGP	27.79%	0.544	0.796
BSGP	**26.97%**	**0.530**	**0.808**

rest is kept for testing. Table 6.3 reports the final test performance, showing that BSGP outperforms the competing methods. Interestingly, in both these large scale experiments, SGHMC-GP always falls back considerably w.r.t. BSGP and even SVGP. We argue that, with these large sized datasets, the continuous alternation of optimization of $\mathbf{Z}$ and $\boldsymbol{\theta}$ and sampling of $\mathbf{u}$ used by the authors (called Moving Window MCEM, see Havasi et al. (2018) for details) might have led to suboptimal solutions.

6.5 CONCLUDING DISCUSSION

We have developed a fully Bayesian treatment of sparse Gaussian process models that considers the inducing inputs, along with the inducing variables and covariance hyper-parameters, as random variables, places suitable priors and carries out approximate inference over them. Our approach, based on SGHMC, investigated two conventional priors (Gaussian and uniform) for the inducing inputs as well as two point process based priors (the Determinantal and the Strauss processes).

By challenging the standard belief of most previous work on sparse GP inference that assumes the inducing inputs can be estimated point-wisely, we have developed a state-of-the-art inference method and have demonstrated its outstanding performance on both accuracy and running time on regression and classification problems. We hope this work can have an impact similar (or better) to other works in machine learning that have adopted more elaborate Bayesian machinery (e.g. Wallach et al., 2009) for long-standing inference problems in commonly used probabilistic models.

Finally, we believe it is worth investigating further more structured priors similar to those presented here (e.g. exploring different hyper-parameter settings), including a full joint treatment of inducing inputs and their number, i.e. $p(\mathbf{Z}, M)$.

7

FINAL CONSIDERATIONS

In this thesis, we discussed several challenges of using approximate Bayesian inference for deep learning models. Broadly speaking, Bayesian deep learning is hindered by problems related to scalability and quality of the approximation inference. In summary, we analyzed several issues that arises from applying variational inference (VI) to deep neural networks, we studied the role of the prior for improving modeling flexibility and we exploited the connections between neural networks and Gaussian processs (GPS) to refine the model and inference approximations of sparse GPS with inducing points.

7.1 SUMMARY OF THE CONTRIBUTIONS AND OPEN PROBLEMS

This thesis makes contributions which follow four different themes:

- **Initialization of variational inference** [Chapter 3]

  In this chapter we discussed how the role of initialization is severely under reported in the literature of variational inference for Bayesian neural networks (BNNS). The reasons for the failing of variational inference for BNNS in practice might depends heavily on the initialization of the variational parameters. We therefore proposed a novel way to initialize variational parameters, called iterative Bayesian linear modeling (IBLM), which is based on an iterative layer-wise initialization carried out through on Bayesian linear models. Empirical evidence is shown through a series of experiments, including regression and classification with deep neural networks (DNNS) and convolutional neural networks (CNNs) which demonstrated the ability of our approach to consistently initialize the optimization in a way that makes convergence faster than alternatives.

Open questions: recent works have exploited loss-landscape analysis and the model-connectivity of deep networks to—among other things—understand how and why simple Bayesian approximations like deep ensembles work in practice. This methodology could be similarly used for studying the loss landscape of the variational objective, and possibly deriving new initializations.

- **Structured parameterizations of variational posteriors [Chapter 4]**

  Inspired by the literature on scalable kernel methods, this chapter proposed Walsh-Hadamard variational inference (WHVI). WHVI offers a novel parameterization of the variational posterior, which is particularly attractive for over-parameterized models, such as modern DNNs and CNNs. WHVI assumes a matrix-variate posterior distribution, which therefore captures covariances across weights. Crucially, unlike previous works on matrix-variate posteriors for VI, this is achieved with a parsimonious parameterization and fast computations, bypassing the over-regularization issues of VI for over-parameterized models. The large experimental campaign demonstrates that WHVI is a strong competitor with other variational approaches for such models, while offering considerable speedups.

  Open questions: over-parameterization is tightly connected with generalization capabilities of deep models; in fact model performance is affected by the network size with bigger and wider neural networks being more resilient to overfit. At the same time, how this analysis carries over to BNN trained with variational inference is still open to debate.

- **Priors in Bayesian deep learning [Chapter 5]**

  In this chapter we analyzed the problem of selection good priors using two different perspectives: functional priors and empirical Bayes using the Wasserstein distance as a proxy to the marginal likelihood. For the first case, we proposed a novel objective based on the Wasserstein distance, and we showed that this objective offers a tractable and stable way to optimize the priors over model parameters. Starting from samples of the neural networks prior, this is done by minimizing their distances to tractable functional priors (e.g.

a Gaussian process), effectively optimizing the priors over model parameters so as to reflect these functional specifications. Alternatively, we can optimize the prior parameters akin to model selection for Gaussian process via marginal likelihood maximization. For this, we derived a practical and efficient optimization framework, based on the minimization of the distributional sliced-Wasserstein distance between the distribution induced by the model and the data generating distribution.

Open questions: the main open question is how we can choose the target stochastic process to map BNN priors to. In principle, this can be generalized, for example, to match deep Gaussian processs (DGPS) or we could use a different prior parameterization to also enforce other behaviors, like e.g. sparsity. On the other hand, we can also debate whether there could be better alternatives to empirical Bayes with marginal likelihood maximization for model selection.

- **Scalable approximations for (deep) Gaussian processes [Chapter 6]**

  In this chapter we proposed a fully Bayesian treatment of sparse Gaussian process models by Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) sampling of the Gaussian process posterior with respect to all hyperparameters, inducing inputs and also inducing locations. In fact, despite significant insights with regards to the benefits of full Bayesian inference over latent variables in GP models, the common practice is to optimize the inducing inputs. The Bayesian treatment considers a wider family of data supporting hypotheses, especially in terms of inducing locations whose marginal support we can represent more accurately. In essence, we showed that, by revisiting old model approximations such as the fully independent training conditional (FITC) endowed with powerful sampling-based inference methods, treating both inducing locations and GP hyper-parameters in a Bayesian way can improve performance significantly.

  Open questions: it is possible to investigate further more structured priors on the inducing locations (e.g. exploring different hyper-parameter settings), including a full joint treatment of inducing inputs and their number. Alternatively, this framework could be in principle extended to inter-domain Gaussian processes with inducing variables, which include—among others—the (deep) convolutional Gaussian process.

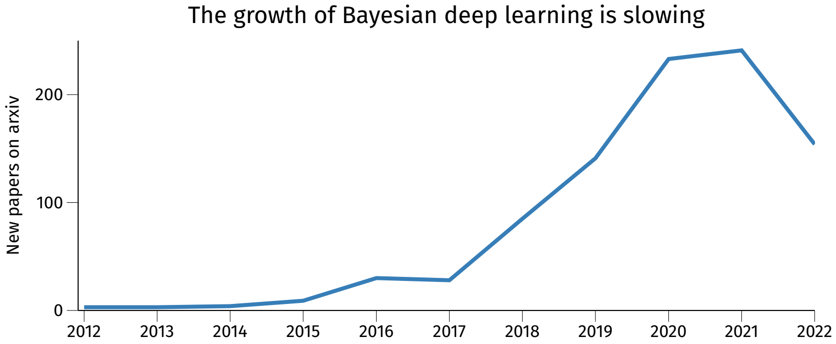


Figure 7.1: Number of unique new papers submitted on arxiv with title or abstract containing different permutations of *Bayesian deep learning*. Data crawled from the public API on December 8th 2021.

7.2 IS BAYESIAN DEEP LEARNING SOLVED? AND NOW WHAT?

We conclude the thesis with a digression and speculations on the future of Bayesian deep learning in general. Looking back four years at the beginning of this thesis, we can better appreciate the rapid and prolific evolution of Bayesian deep learning (Figure 7.1). By late 2017 when this thesis started (at least as a preliminary discussion with Prof. Filippone), not many works were showing practical inference of Bayesian neural networks to succeed beyond one hidden layer and few hidden units, let alone for more complex architectures like large scale CNNs. In the span of a couple of years this changed, thanks to an increased interest in scalable probabilistic deep models as well as to the democratization of flexible libraries (like Tensorflow and PyTorch) and powerful computational devices (like GPUs and TPUs). This growth is even more amazing, considering that it originated concurrently for different Bayesian approximations, whether it is variational, sampling or ensembles.

We have reached a point now which makes us wonder if we can consider Bayesian deep learning methodologically solved. While this sounds very provocative, it is important to remember that the field as a whole evolved so quickly that what could not have been done four years ago is today possible. Whether it is variational inference, or Laplace approximation, or ensembles or scalable Markov-Chain Monte Carlo sampling, we have significant empirical evidence and knowledge supporting whichever method. Practitioners have today a tool-set of countless probabilistic methods to choose from, with different models being more suitable than others for different application scenarios. Yet, with only few exceptions, Bayesian deep learning is still rarely employed for end-to-end applications, despite clear benefits for uncertainty quantification and calibration. Arguably, much of the deep learning success can be attributed to simple prototyping and simple deployment and monitoring of the models. While new trends are emerging in this direction (Novak et al., 2020; E. Daxberger et al., 2021), this is generally still out of reach for Bayesian deep learning. With this thesis, we have given the practitioners a series of new tools and suggestions to make Bayesian deep learning work in practice, with the hope that applied research could benefit from it.

From a different point of view, we can also question how much (Bayesian) deep learning is sustainable and truly scalable in the long term. In a recent survey on the future role of deep learning for discovery of new fundamental physics (Karagiorgi et al., 2021), the Authors project that in order to guarantee statistical significance for new discoveries, the model and the overall data-processing system should be able to keep up with data streams of at least 40 Tbit/s. This is already embarrassingly problematic for deep learning (for comparison, it is equivalent to roughly 30 ImageNet *every second*), let alone for Bayesian inference, to the point that we might need to rethink the entire computational pipeline. Uncertainty is omnipresent in nature (Heisenberg, 1927; Kennard, 1927) and yet our computational devices are for the most part deterministic. On a long term horizon, quantum devices could ease the burden of probabilistic inference (Zhao et al., 2019; Schuld and Petruccione, 2021; Huang et al., 2021). Also, while until now Bayesian deep learning has been chasing deep learning in the race for bigger models and new architectures, with quantum computing Bayesian inference could lead deep learning—and machine learning in general—to novel and exciting research directions. This also includes the possibility to redefine our objectives. This thesis, among countless of other examples in the literature, praised the elegance of Bayes' rule for updating beliefs with evidence, but Bayesian statistics should not be considered as an immovable dogma (Popper, 1934). For example, even without entering in the philosophical debate of "frequestist" versus "Bayesian", it is possible to generalize Bayes' theorem in such a way that inference is carried out on bounds of probability measures (Dempster, 1966; Dempster, 1968) or to unify and derive new methods from a common root of learning algorithms (M. E. Khan and Rue, 2021).

BIBLIOGRAPHY

Abadi, M. et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org (cit. on pp. 3, 26).

Acciarri, R. et al. (2017). “Convolutional neural networks applied to neutrino events in a liquid argon time projection chamber.” In: *Journal of Instrumentation* 12.3 (cit. on p. 3).

Akaike, H. (1973). “Information Theory and an Extension of the Maximum Likelihood Principle.” In: *2nd International Symposium on Information Theory, 1973*. Publishing House of the Hungarian Academy of Sciences, pp. 268–281 (cit. on p. 96).

Amit, R. and R. Meir (2018). “Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory.” In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*. Vol. 80. PMLR, pp. 205–214 (cit. on p. 84).

Arjovsky, M., S. Chintala, and L. Bottou (2017). “Wasserstein Generative Adversarial Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 214–223 (cit. on p. 86).

Ashukha, A., A. Lyzhov, D. Molchanov, and D. Vetrov (2020). “Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning.” In: *International Conference on Learning Representations* (cit. on p. 91).

Atanov, A., A. Ashukha, K. Struminsky, D. Vetrov, and M. Welling (2019). “The Deep Weight Prior.” In: *International Conference on Learning Representations* (cit. on p. 84).

Aurisano, A. et al. (2016). “A convolutional neural network neutrino event classifier.” In: *Journal of Instrumentation* 11.9 (cit. on p. 3).

Baldi, P., P. Sadowski, and D. Whiteson (2014). “Searching for Exotic Particles in High-Energy Physics with Deep Learning.” In: *Nature Communications* 5.1, p. 4308 (cit. on p. 128).

Baldi, P. and K. Hornik (1989). “Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima.” In: *Neural Networks* 2.1, pp. 53–58 (cit. on p. 36).

Barber, D. and C. Bishop (1998). “Ensemble learning in Bayesian neural networks.” In: *Generalization in Neural Networks and Machine Learning*. Springer Verlag, pp. 215–237 (cit. on p. 9).

Barber, D. and C. K. I. Williams (1997). “Gaussian Processes for Bayesian Classification via Hybrid Monte Carlo.” In: *Advances in Neural Information Processing Systems 9*. MIT Press, pp. 340–346 (cit. on p. 108).

Battiti, R. (1989). “Accelerated Backpropagation Learning: Two Optimization Methods.” In: *Complex Systems* 3.4, pp. 331–342 (cit. on p. 2).

Bauer, M., M. van der Wilk, and C. E. Rasmussen (2016). “Understanding Probabilistic Sparse Gaussian Process Approximations.” In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 1533–1541 (cit. on pp. 111, 112).

Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind (2017). “Automatic Differentiation in Machine Learning: A Survey.” In: *The Journal of Machine Learning Research* 18, pp. 5595–5637 (cit. on p. 3).

Becker, S. and Y. LeCun (1989). “Improving the Convergence of Back-Propagation Learning with Second-Order Methods.” In: *Proc. of the 1988 Connectionist Models Summer School*. Morgan Kaufman, pp. 29–37 (cit. on p. 2).

Bellec, G., D. Salaj, A. Subramoney, R. Legenstein, and W. Maass (2018). “Long short-term memory and learning-to-learn in networks of spiking neurons.” In: *Advances in Neural Information Processing Systems*, pp. 787–797 (cit. on p. 3).

Bengio, Y., A. Courville, and P. Vincent (2013). “Representation Learning: A Review and New Perspectives.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1798–1828 (cit. on p. 100).

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2006). “Greedy Layer-Wise Training of Deep Networks.” In: *Advances in Neural Information Processing Systems 19*. MIT Press, pp. 153–160 (cit. on pp. 35, 36).

Betancourt, M. (2015). “The Fundamental Incompatibility of Scalable Hamiltonian Monte Carlo and Naive Data Subsampling.” In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. JMLR.org, pp. 533–540 (cit. on p. 10).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. 1st ed. 2006. Corr. 2nd printing 2011. Springer (cit. on pp. 18, 34, 38).

Blei, D. M., M. I. Jordan, and J. W. Paisley (2012). “Variational Bayesian Inference with Stochastic Search.” In: *Proceedings of the 29th International Conference on Machine Learning*. ACM, pp. 1367–1374 (cit. on p. 25).

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017). “Variational Inference: A Review for Statisticians.” In: 112.518, pp. 859–877 (cit. on p. 22).

Blei, D., S. Mohamed, and R. Ranganath (2016). “Variational Inference: Foundations and Modern Methods.” In: *Neural Information Processing Systems (Tutorial)* (cit. on p. 23).

Blundell, C., J. Cornebise, K. Kavukcuoglu, and D. Wierstra (2015). “Weight Uncertainty in Neural Network.” In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. PMLR, pp. 1613–1622 (cit. on pp. 9, 76, 82).

Bojarski, M. et al. (2017). “Structured Adaptive and Random Spinners for Fast Machine Learning Computations.” In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Vol. 54. PMLR, pp. 1020–1029 (cit. on pp. 57, 76).

Bonilla, E. V., K. Krauth, and A. Dezfouli (2019). “Generic Inference in Latent Gaussian Process Models.” In: *Journal of Machine Learning Research* 20.117, pp. 1–63 (cit. on p. 108).

Bowman, S. R., L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio (2016). “Generating Sentences from a Continuous Space.” In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pp. 10–21 (cit. on pp. 50, 54, 76).

Bradbury, J. et al. (2018). *JAX: Composable Transformations of Python+NumPy Programs*. Version 0.2.5 (cit. on p. 3).

Briol, F.-X., C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic (2019). “Probabilistic Integration: A Role in Statistical Computation?” In: *Statistical Science* 34.1, pp. 1–22 (cit. on p. 79).

Bui, T. D., J. Yan, and R. E. Turner (2017). “A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation.” In: *Journal of Machine Learning Research* 18.1, pp. 3649–3720 (cit. on p. 111).

Burgess, C. P., I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner (2018). “Understanding disentangling in β-VAE.” In: *CoRR* abs/1804.03599 (cit. on p. 76).

Chan, J., A. C. Miller, and E. B. Fox (2020). “Representing and Denoising Wearable ECG Recordings.” In: *CoRR* abs/2012.00110 (cit. on p. 1).

Chellapilla, K., S. Puri, and P. Simard (2006). “High Performance Convolutional Neural Networks for Document Processing.” In: *International Workshop on Frontiers in Handwriting Recognition* (cit. on p. 2).

Chen, T., E. Fox, and C. Guestrin (2014). “Stochastic Gradient Hamiltonian Monte Carlo.” In: *Proceedings of the 31st International Conference on Machine Learning*. PMLR, pp. 1683–1691 (cit. on pp. 10, 30, 31, 67, 82, 83, 101, 104, 112, 117).

Chen, X., D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel (2017). “Variational Lossy Autoencoder.” In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net (cit. on p. 90).

Cho, Y. and L. K. Saul (2009). “Kernel Methods for Deep Learning.” In: *Advances in Neural Information Processing Systems* 22. Curran Associates, Inc., pp. 342–350 (cit. on p. 56).

Ciresan, D. C., U. Meier, and J. Schmidhuber (2012). “Multi-column deep neural networks for image classification.” In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 3642–3649 (cit. on p. 2).

Cockayne, J., C. J. Oates, I. C. Ipsen, and M. Girolami (2019). “A Bayesian Conjugate Gradient Method (with Discussion).” In: *Bayesian Analysis* 14.3, pp. 937–1012 (cit. on p. 79).

Cottrell, G. W., P. Munro, and D. Zipser (1989). “Image Compression by Back Propagation: A Demonstration of Extensional Programming.” In: *Models of Cognition*, pp. 208–240 (cit. on p. 100).

Cutajar, K., E. V. Bonilla, P. Michiardi, and M. Filippone (2017). “Random Feature Expansions for Deep Gaussian Processes.” In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 884–893 (cit. on pp. 76, 108).

Daley, D. J. and D. Vere-Jones (2003). *An introduction to the theory of point processes. Vol. I.* Second. Elementary theory and methods. Springer-Verlag (cit. on p. 118).

Damianou, A. C. and N. D. Lawrence (2013). “Deep Gaussian Processes.” In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013*. Vol. 31. JMLR.org, pp. 207–215 (cit. on pp. 119, 120).

Dang, V. N. et al. (2022). “Vessel-CAPTCHA: An efficient learning framework for vessel annotation and segmentation.” In: *Medical Image Analysis* 75 (cit. on p. 1).

Dangel, F., F. Kunstner, and P. Hennig (2020). “BackPACK: Packing more into Backprop.” In: *Proceedings of the 8th International Conference on Learning Representations*. OpenReview.net (cit. on p. 10).

Daxberger, E. A., E. T. Nalisnick, J. U. Allingham, J. Antorán, and J. M. Hernández-Lobato (2021). “Bayesian Deep Learning via Subnetwork Inference.” In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, pp. 2510–2521 (cit. on p. 10).

Daxberger, E., A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig (2021). “Laplace Redux - Effortless Bayesian Deep Learning.” In: *CoRR* abs/2106.14806 (cit. on p. 135).

DeGroot, M. H. and S. E. Fienberg (1983). “The Comparison and Evaluation of Forecasters.” In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 32.1/2, pp. 12–22 (cit. on p. 48).

Dempster, A. P. (1966). “New Methods for Reasoning Towards Posterior Distributions Based on Sample Data.” In: *The Annals of Mathematical Statistics* 37.2, pp. 355–374 (cit. on p. 135).

Dempster, A. P. (1968). “A Generalization of Bayesian Inference.” In: *Journal of the Royal Statistical Society: Series B (Methodological)* 30.2, pp. 205–232 (cit. on p. 135).

Denker, J. S. and Y. LeCun (1990). “Transforming Neural-Net Output Levels to Probability Distributions.” In: *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, pp. 853–859 (cit. on p. 8).

Denker, J., D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield (1987). “Large automatic learning, rule extraction, and generalization.” English. In: *Complex Systems* 1.5, pp. 877–922 (cit. on p. 8).

Dieleman, S., K. W. Willett, and J. Dambre (2015). “Rotation-invariant convolutional neural networks for galaxy morphology prediction.” In: *Monthly Notices of the Royal Astronomical Society* 450.2, pp. 1441–1459 (cit. on p. 3).

Dieng, A. B., D. Tran, R. Ranganath, J. W. Paisley, and D. M. Blei (2017). “Variational Inference via \chi Upper Bound Minimization.” In: *Advances in Neural Information Processing Systems 30*, pp. 2732–2741 (cit. on p. 23).

Ding, S. and D. Cook (2014). “Dimension folding PCA and PFC for matrix-valued predictors.” In: *Statistica Sinica* 24.1, pp. 463–492 (cit. on p. 59).

Du, M., F. Yang, N. Zou, and X. Hu (2021). “Fairness in Deep Learning: A Computational Perspective.” In: *IEEE Intelligent Systems* 36.4, pp. 25–34 (cit. on p. 1).

Duane, S., A. Kennedy, B. J. Pendleton, and D. Roweth (1987). “Hybrid Monte Carlo.” In: *Physics Letters B* 195.2, pp. 216–222 (cit. on p. 9).

Duchi, J., E. Hazan, and Y. Singer (2011). “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.” In: *Journal of Machine Learning Research* 12, pp. 2121–2159 (cit. on pp. 2, 31, 35, 37).

Dunlop, M. M., M. A. Girolami, A. M. Stuart, and A. L. Teckentrup (2018). “How Deep Are Deep Gaussian Processes?” In: *Journal of Machine Learning Research* 19.1, pp. 2100–2145 (cit. on p. 76).

Duvenaud, D. K., O. Rippel, R. P. Adams, and Z. Ghahramani (2014). “Avoiding pathologies in very deep networks.” In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. Vol. 33. JMLR.org, pp. 202–210 (cit. on pp. 76, 81, 104).

Dziugaite, G. K. and D. M. Roy (2017). “Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data.” In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017* (cit. on p. 35).

Edwards, A. W. F. (1978). “Commentary on the Arguments of Thomas Bayes.” In: *Scandinavian Journal of Statistics* 5.2, pp. 116–118 (cit. on p. 15).

Erhan, D., A. Courville, and P. Vincent (2010). "Why Does Unsupervised Pre-training Help Deep Learning?" In: *Journal of Machine Learning Research* 11, pp. 625–660 (cit. on p. 36).

Farquhar, S., L. Smith, and Y. Gal (2020). "Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations." In: *Advances in Neural Information Processing Systems 33* (cit. on p. 10).

Filippone, M. and M. Girolami (2014). "Pseudo-Marginal Bayesian Inference for Gaussian Processes." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11, pp. 2214–2226 (cit. on pp. 108, 119).

Fino and Algazi (1976). "Unified Matrix Treatment of the Fast Walsh-Hadamard Transform." In: *IEEE Transactions on Computers* C-25.11, pp. 1142–1146 (cit. on p. 57).

Flach, P. A. (2016). "Classifier Calibration." In: *Encyclopedia of Machine Learning and Data Mining*. Springer US, pp. 1–8 (cit. on p. 48).

Flam-Shepherd, D., J. Requeima, and D. Duvenaud (2017). "Mapping Gaussian Process Priors to Bayesian Neural Networks." In: *NeurIPS workshop on Bayesian Deep Learning* (cit. on pp. 85, 97, 104).

Flam-Shepherd, D., J. Requeima, and D. Duvenaud (2018). "Characterizing and Warping the Function space of Bayesian Neural Networks." In: *NeurIPS workshop on Bayesian Deep Learning* (cit. on pp. 85, 104).

Franzese, G., D. Milios, M. Filippone, and P. Michiardi (2021a). "A Scalable Bayesian Sampling Method Based on Stochastic Gradient Descent Isotropization." In: *Entropy* 23.11 (cit. on p. 10).

Franzese, G., D. Milios, M. Filippone, and P. Michiardi (2021b). "Revisiting the effects of stochasticity for Hamiltonian samplers." In: *CoRR* abs/2106.16200 (cit. on p. 11).

Fussell, L. and B. Moews (2019). "Forging new worlds: High-resolution synthetic galaxies with chained generative adversarial networks." In: *Monthly Notices of the Royal Astronomical Society* 485.3, pp. 3215–3223 (cit. on p. 3).

G. Matthews, A. G. de, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani (2018). "Gaussian Process Behaviour in Wide Deep Neural Networks." In: *Proceedings of the 6th International Conference on Learning Representations* (cit. on pp. 21, 76, 81).

Gal, Y. (2016). "Uncertainty in Deep Learning." Doctoral dissertation. University of Cambridge (cit. on p. 8).

Gal, Y. and Z. Ghahramani (2016a). *Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference*. arXiv:1506.02158 (cit. on pp. 34, 35, 47, 76).

Gal, Y. and Z. Ghahramani (2016b). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of The 33rd International Conference* *on Machine Learning*. Vol. 48. PMLR, pp. 1050–1059 (cit. on pp. 9, 34, 43, 54, 55, 67, 76, 108).

Gal, Y., J. Hron, and A. Kendall (2017). "Concrete Dropout." In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 3581–3590 (cit. on p. 34).

Garipov, T., P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson (2018). "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs." In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., pp. 8789–8798 (cit. on pp. 11, 35).

Garriga-Alonso, A., C. E. Rasmussen, and L. Aitchison (2019). "Deep Convolutional Networks as shallow Gaussian Processes." In: *International Conference on Learning Representations* (cit. on pp. 21, 76).

Gauss, C. F. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium* (cit. on p. 2).

Gauss, C. F. (1821). *Theoria combinationis observationum erroribus minimis obnoxiae (Theory of the combination of observations least subject to error)* (cit. on p. 2).

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2004). *Bayesian Data Analysis*. 2nd ed. Chapman and Hall/CRC (cit. on pp. 67, 128).

Ghahramani, Z. (2013). "Bayesian non-parametrics and the probabilistic approach to modelling." In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1984 (cit. on p. 16).

Ghahramani, Z. (2015). "Probabilistic Machine Learning and Artificial Intelligence." In: *Nature* 521.7553, pp. 452–459 (cit. on pp. 16, 34).

Giraldo, J.-J. and M. A. Álvarez (2019). "A Fully Natural Gradient Scheme for Improving Inference of the Heterogeneous Multi-Output Gaussian Process Model." In: *arXiv preprint arXiv:1911.10225* (cit. on p. 109).

Glorot, X. and Y. Bengio (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. PMLR, pp. 249–256 (cit. on pp. 35, 36, 43).

Gonzalez, J. A., F. J. Rodriguez-Cortes, O. Cronie, and J. Mateu (2016). "Spatio-temporal Point Process Statistics: A Review." In: *Spatial Statistics* 18, pp. 505–544 (cit. on p. 117).

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc. (cit. on p. 86).

Graves, A. (2011). "Practical Variational Inference for Neural Networks." In: *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., pp. 2348–2356 (cit. on pp. 9, 27, 34, 35, 37, 54, 76, 82).

Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola (2012). "A Kernel Two-sample Test." In: *Journal of Machine Learning Research* 13, pp. 723–773 (cit. on p. 56).

Gretton, A., K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola (2008). "A Kernel Statistical Test of Independence." In: *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., pp. 585–592 (cit. on p. 56).

Grigorescu, S. M., B. Trasnea, T. T. Cocias, and G. Macesanu (2020). "A survey of deep learning techniques for autonomous driving." In: *Journal of Field Robotics* 37.3, pp. 362–386 (cit. on p. 1).

Grover, A., M. Dhar, and S. Ermon (2018). "Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models." In: *Proceedings of the 32nd Conference on Artificial Intelligence, AAAI 2018*. AAAI Press, pp. 3069–3076 (cit. on p. 90).

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville (2017). "Improved Training of Wasserstein GANs." In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., pp. 5767–5777 (cit. on p. 87).

Guo, C., G. Pleiss, Y. Sun, and K. Q. Weinberger (2017). "On Calibration of Modern Neural Networks." In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 1321–1330 (cit. on pp. 1, 33, 48).

Gupta, A. K. and D. K. Nagar (1999). *Matrix variate distributions*. Chapman and Hall/CRC (cit. on p. 62).

Ha, D., A. M. Dai, and Q. V. Le (2016). *HyperNetworks*. arXiv:1609.09106 (cit. on p. 34).

Ha, D., A. M. Dai, and Q. V. Le (2017). "HyperNetworks." In: *International Conference on Learning Representations* (cit. on p. 85).

Hadamard, J. (1908). *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Imprimerie nationale (cit. on p. 2).

Hafner, D., D. Tran, T. P. Lillicrap, A. Irpan, and J. Davidson (2019). "Noise Contrastive Priors for Functional Uncertainty." In: *Proceedings of the 35h Conference on Uncertainty in Artificial Intelligence, UAI 2019*. AUAI Press, p. 332 (cit. on p. 84).

Havasi, M., J. M. Hernández-Lobato, and J. J. Murillo-Fuentes (2018). "Inference in Deep Gaussian Processes using Stochastic Gradient Hamiltonian Monte Carlo." In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., pp. 7506–7516 (cit. on pp. 108, 117, 122, 123, 125, 129).

He, B., B. Lakshminarayanan, and Y. W. Teh (2020). "Bayesian Deep Ensembles via the Neural Tangent Kernel." In: *Advances in Neural Information Processing Systems 33* (cit. on p. 11).

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778 (cit. on pp. 35, 39, 70, 93).

Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York (cit. on p. 2).

Heisenberg, W. (1927). "Uber den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik." In: *Zeitschrift fur Physik* 43.3-4, pp. 172–198 (cit. on p. 135).

Helgason, S. (2010). *Integral Geometry and Radon Transforms*. Springer Science & Business Media (cit. on p. 98).

Hensman, J., N. Fusi, and N. D. Lawrence (2013). "Gaussian Processes for Big Data." In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pp. 282–290 (cit. on pp. 108, 122).

Hensman, J., A. G. Matthews, M. Filippone, and Z. Ghahramani (2015). "MCMC for Variationally Sparse Gaussian Processes." In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 1648–1656 (cit. on pp. 74, 108, 109, 111, 112, 121, 123, 124, 128).

Hensman, J., A. Matthews, and Z. Ghahramani (2015). "Scalable Variational Gaussian Process Classification." In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015*. Vol. 38. PMLR, pp. 351–360 (cit. on pp. 108, 123, 125).

Hernandez-Lobato, J. M. and R. Adams (2015). "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks." In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. PMLR, pp. 1861–1869 (cit. on pp. 9, 68, 76).

Higgins, I., L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: *International Conference on Learning Representations* (cit. on p. 76).

Hinton, G. E. and D. van Camp (1993). "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights." In: *Proceedings of the Sixth Annual Conference on Computational Learning Theory* (cit. on pp. 9, 24).

Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks." In: *Science* 313.5786, pp. 504–507 (cit. on p. 100).

Hochreiter, S. and J. Schmidhuber (1997). "Long Short-Term Memory." In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 2).

Hoffman, M. D., D. M. Blei, C. Wang, and J. W. Paisley (2013). "Stochastic Variational Inference." In: *Journal of Machine Learning Research* 14.1, pp. 1303–1347 (cit. on p. 27).

Huang, H.-Y., M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean (2021). "Power of data in quantum machine learning." In: *Nature Communications* 12.1, pp. 1–9 (cit. on p. 135).

Hubara, I., M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio (2016). "Binarized Neural Networks." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 4107–4115 (cit. on p. 76).

Immer, A., M. Bauer, V. Fortuin, G. Rätsch, and M. E. Khan (2021). "Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning." In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, pp. 4563–4573 (cit. on pp. 10, 82, 96).

Immer, A., M. Korzepa, and M. Bauer (2021). "Improving predictions of Bayesian neural nets via local linearization." In: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. PMLR, pp. 703–711 (cit. on pp. 10, 84).

Izmailov, P., S. Vikram, M. D. Hoffman, and A. G. Wilson (2021). "What Are Bayesian Neural Network Posteriors Really Like?" In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, pp. 4629–4640 (cit. on pp. 11, 104).

Jacobs, R. A. (1988). "Increased rates of convergence through learning rate adaptation." In: *Neural Networks* 1.4, pp. 295–307 (cit. on p. 2).

Jang, P. A., A. Loeb, M. Davidow, and A. G. Wilson (2017). "Scalable Lévy Process Priors for Spectral Kernel Learning." In: *Advances in Neural Information Processing Systems 30*, pp. 3940–3949 (cit. on p. 116).

Jeffreys, H. (1939). *Theory of Probability*. Oxford (cit. on p. 15).

Jeffreys, H. (1946). "An invariant form for the prior probability in estimation problems." In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 186.1007, pp. 453–461 (cit. on p. 16).

Jia, Y. (2014). "Learning Semantic Image Representations at a Large Scale." Doctoral dissertation. EECS Department, University of California, Berkeley (cit. on p. 41).

Jiménez-Luna, J., F. Grisoni, and G. Schneider (1, 2020). "Drug discovery with explainable artificial intelligence." In: *Nature Machine Intelligence*, pp. 573–584 (cit. on p. 1).

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1, 1999). "An Introduction to Variational Methods for Graphical Models." In: *Machine Learning* 37.2, pp. 183–233 (cit. on pp. 22, 34, 53).

Jumper, J. et al. (2021). "Highly accurate protein structure prediction with AlphaFold." In: *Nature* 596.7873, pp. 583–589 (cit. on p. 1).

Kantorovich, L. V. (1942). "On the transfer of masses." In: *Doklady Akademii Nauk SSSR* 37, pp. 227–229 (cit. on p. 86).

Kantorovich, L. V. (1948). "On a problem of Monge." In: *Uspekhi Matematicheskikh Nauk* 3, pp. 225–226 (cit. on p. 86).

Karagiorgi, G., G. Kasieczka, S. Kravitz, B. Nachman, and D. Shih (2021). "Machine Learning in the Search for New Fundamental Physics." In: (cit. on p. 135).

Karaletsos, T. and T. Bui (2019). "Gaussian Process Meta-Representations For Hierarchical Neural Network Weight Priors." In: *2nd Symposium on Advances in Approximate Bayesian Inference* (cit. on p. 84).

Karaletsos, T. and T. Bui (2020). "Hierarchical Gaussian Process Priors for Bayesian Neural Network Weights." In: *Advances in Neural Information Processing Systems*. Vol. 33 (to appear) (cit. on p. 84).

Kendall, A. and Y. Gal (2017). "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 5574–5584 (cit. on pp. 9, 33, 79).

Kendall, A., J. Hawke, et al. (2019). "Learning to Drive in a Day." In: *Proceedings of the International Conference on Robotics and Automation*. IEEE, pp. 8248–8254 (cit. on p. 1).

Kennard, E. H. (1927). "Zur Quantenmechanik einfacher Bewegungstypen." In: *Zeitschrift fur Physik* 44.4-5, pp. 326–352 (cit. on p. 135).

Khan, M. E. E., A. Immer, E. Abedi, and M. Korzepa (2019). "Approximate Inference Turns Deep Networks into Gaussian Processes." In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc. (cit. on pp. 21, 84).

Khan, M. E., D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava (2018). "Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam." In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, pp. 2616–2625 (cit. on p. 9).

Khan, M. E. and H. Rue (2021). "The Bayesian Learning Rule." In: *CoRR* abs/2107.04562 (cit. on p. 135).

Kim, Y., C. Yang, Y. Kim, G. X. Gu, and S. Ryu (2020). "Designing an Adhesive Pillar Shape with Deep Learning-Based Optimization." In: *ACS Applied Materials and Interfaces* 12.21, pp. 24458–24465 (cit. on p. 3).

Kingma, D. P. and J. Ba (2015). "Adam: A Method for Stochastic Optimization." In: *Proceedings of the 3rd International Conference on Learning Representations* (cit. on pp. 2, 28, 35, 37, 44, 62, 66, 91, 122).

Kingma, D. P. and M. Welling (2014). "Auto-Encoding Variational Bayes." In: *Proceedings of the 2nd International Conference on Learning Representations* (cit. on pp. 25, 34, 37, 62, 100).

Kingma, D. P. and M. Welling (2019). "An Introduction to Variational Autoencoders." In: *Foundations and Trends in Machine Learning* 12.4, pp. 307–392 (cit. on p. 101).

Kingma, D. P., T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling (2016). "Improved Variational Inference with Inverse Autoregressive Flow." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 4743–4751 (cit. on pp. 66, 70, 90, 164).

Kiran, B. R., I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez (2021). "Deep Reinforcement Learning for Autonomous Driving: A Survey." In: *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18 (cit. on p. 1).

Kolouri, S., K. Nadjahi, U. Simsekli, R. Badeau, and G. K. Rohde (2019). "Generalized Sliced Wasserstein Distances." In: *Advances in Neural Information Processing Systems 32*, pp. 261–272 (cit. on p. 98).

Kraska, T., A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis (2018). "The Case for Learned Index Structures." In: *Proceedings of the 2018 International Conference on Management of Data*. Association for Computing Machinery, pp. 489–504 (cit. on p. 3).

Krauth, K., E. V. Bonilla, K. Cutajar, and M. Filippone (2017). "AutoGP: Exploring the Capabilities and Limitations of Gaussian Process Models." In: *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017*. AUAI Press (cit. on p. 108).

Krishnan, R., D. Liang, and M. Hoffman (2018). "On the Challenges of Learning with Inference Networks on Sparse, High-Dimensional Data." In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Vol. 84. PMLR, pp. 143–151 (cit. on p. 37).

Kristiadi, A., M. Hein, and P. Hennig (2020). "Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks." In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. PMLR, pp. 5436–5446 (cit. on p. 10).

Krizhevsky, A. and G. Hinton (2009). "Learning multiple layers of features from tiny images." In: *Master's thesis, Department of Computer Science, University of Toronto* (cit. on p. 93).

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc. (cit. on pp. 2, 48, 70).

Kwon, J. and L. P. Carloni (2020). "Transfer learning for design-space exploration with high-level synthesis." In: *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD* (cit. on p. 3).

Lakshminarayanan, B., A. Pritzel, and C. Blundell (2017). "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 6402–6413 (cit. on pp. 11, 33, 91).

Lanusse, F., P. Melchior, and F. Moolekamp (2019). "Hybrid physical-deep learning model for astronomical inverse problems." In: *arXiv* (cit. on p. 3).

Lavancier, F., J. Møller, and E. Rubak (2015). "Determinantal Point Process Models and Statistical Inference." In: *Royal Statistical Society B* 77, pp. 853–877 (cit. on p. 118).

Lawrence, N. D. (2005). "Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models." In: *Journal of Machine Learning Research* 6, pp. 1783–1816 (cit. on p. 100).

Lázaro-Gredilla, M., J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal (2010). "Sparse Spectrum Gaussian Process Regression." In: *Journal of Machine Learning Research* 11, pp. 1865–1881 (cit. on pp. 73, 76).

Lazaro-Gredilla, M. and A. Figueiras-Vidal (2009). "Inter-domain Gaussian Processes for Sparse Inference using Inducing Features." In: *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., pp. 1087–1095 (cit. on p. 108).

Le, Q., T. Sarlos, and A. Smola (2013). "Fastfood - Approximating Kernel Expansions in Loglinear Time." In: *Proceedings of the 30th International Conference on Machine Learning* (cit. on pp. 55, 56, 75, 76).

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition." In: *Neural Computation* 1.4, pp. 541–551 (cit. on p. 2).

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990). "Handwritten digit recognition with a back-propagation network." In: *Advances in Neural Information Processing Systems 2*. Morgan Kaufman (cit. on p. 2).

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-Based Learning Applied to Document Recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 2, 36, 45, 93).

LeCun, Y., P. Y. Simard, and B. A. Pearlmutter (1993). "Automatic Learning Rate Maximization by On-Line Estimation of the Hessian's Eigenvectors." In: *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, pp. 156–163 (cit. on p. 2).

LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep learning." In: *Nature* 521.7553, pp. 436–444 (cit. on pp. 1, 33).

Lee, J., J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri (2018). "Deep Neural Networks as Gaussian Processes." In: *International Conference on Learning Representations* (cit. on p. 21).

Legendre, A. M. (1805). *Nouvelles méthodes pour la détermination des orbites des cometes*. F. Didot (cit. on p. 2).

Leibniz, G. W. (1676). “Memoir using the chain rule (Cited in TMME 7:2&3 p 321-332, 2010).” In: (cit. on p. 2).

Li, G., X. Zhou, S. Li, and B. Gao (2019). “QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning.” In: *Proceedings of the VLDB Endowment* 12.12, pp. 2118–2130 (cit. on p. 3).

Li, Y. and Y. Gal (2017). “Dropout Inference in Bayesian Neural Networks with Alpha-divergences.” In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 2052–2061 (cit. on pp. 9, 23).

Li, Y. and R. E. Turner (2016). “Rényi Divergence Variational Inference.” In: *Advances in Neural Information Processing Systems* 29, pp. 1073–1081 (cit. on p. 23).

Linnainmaa, S. (1970). “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.” Master’s thesis. Univ. Helsinki (cit. on p. 2).

Liu, H., M. Xu, Z. Yu, V. Corvinelli, and C. Zuzarte (2015). “Cardinality Estimation Using Neural Networks.” In: *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., pp. 53–59 (cit. on p. 3).

Liu, Q. and D. Wang (2016). “Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm.” In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 2378–2386 (cit. on p. 9).

Loaiza-Ganem, G. and J. P. Cunningham (2019). “The Continuous Bernoulli: Fixing a Pervasive Error in Variational Autoencoders.” In: *Advances in Neural Information Processing Systems* 32, pp. 13266–13276 (cit. on p. 102).

Louizos, C., K. Ullrich, and M. Welling (2017). “Bayesian Compression for Deep Learning.” In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 3288–3298 (cit. on pp. 54, 76).

Louizos, C. and M. Welling (2016). “Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors.” In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. PMLR, pp. 1708–1716 (cit. on pp. 9, 40, 55, 58, 64, 68).

Louizos, C. and M. Welling (2017). “Multiplicative Normalizing Flows for Variational Bayesian Neural Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 2218–2227 (cit. on pp. 66, 70, 90, 164).

Lu, Y. C., S. S. Kiran Pentapati, L. Zhu, K. Samadi, and S. K. Lim (2020). “TP-GNN: A graph neural network framework for tier partitioning in monolithic 3D ICs.” In: *Proceedings - Design Automation Conference* (cit. on p. 3).

Ma, C., Y. Li, and J. M. Hernández-Lobato (2019). "Variational Implicit Processes." In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, pp. 4222–4233 (cit. on p. 84).

Mackay, D. J. C. (1994). "Bayesian methods for backpropagation networks." In: *Models of Neural Networks III*. Springer. Chap. 6, pp. 211–254 (cit. on pp. 54, 119).

MacKay, D. J. C. (1991). "Bayesian Model Comparison and Backprop Nets." In: *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, pp. 839–846 (cit. on p. 8).

MacKay, D. J. C. (1992). "A Practical Bayesian Framework for Backpropagation Networks." In: *Neural Computation* 4.3, pp. 448–472 (cit. on pp. 8, 17, 103).

MacKay, D. J. C. (1998). "Choice of Basis for Laplace Approximation." In: *Machine Learning* 33.1, pp. 77–86 (cit. on p. 10).

MacKay, D. J. (1995). "Probable Networks and Plausible Predictions - a Review of Practical Bayesian Methods for Supervised Neural Networks." In: *Network: computation in neural systems* 6.3, p. 469 (cit. on p. 82).

MacKay, D. J. and M. N. Gibbs (1999). "Density Networks." In: *Statistics and Neural Networks: Advances at the Interface*, pp. 129–144 (cit. on p. 100).

Maddison, C. J., A. Mnih, and Y. W. Teh (2017). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables." In: *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net (cit. on p. 26).

Marcus, R. and O. Papaemmanouil (2019). "Plan-Structured Deep Neural Network Models for Query Performance Prediction." In: *Proceedings of the VLDB Endowment*. Vol. 12. 11, pp. 1733–1746 (cit. on p. 3).

Matthews, A. G., M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman (2017). "GPflow: A Gaussian Process Library using TensorFlow." In: *Journal of Machine Learning Research* 18.40, pp. 1–6 (cit. on pp. 108, 124).

Metropolis, N. and S. Ulam (1949). "The Monte Carlo Method." In: *Journal of the American Statistical Association* 44.247. PMID: 18139350, pp. 335–341 (cit. on p. 24).

Milios, D., R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone (2018). "Dirichlet-based Gaussian Processes for Large-scale Calibrated Classification." In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., pp. 6008–6018 (cit. on pp. 40, 41).

Milios, D., P. Michiardi, and M. Filippone (2020). "A Variational View on Bootstrap Ensembles as Bayesian Inference." In: *CoRR* abs/2006.04548 (cit. on p. 11).

Mishkin, D. and J. Matas (2016). "All You Need is a Good Init." In: *CoRR* abs/1511.06422 (cit. on pp. 35, 37, 43).

Močkus, J. (1975). “On Bayesian Methods for Seeking the Extremum.” In: *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*. Springer Berlin Heidelberg, pp. 400–404 (cit. on p. 95).

Molchanov, D., A. Ashukha, and D. Vetrov (2017). “Variational Dropout Sparsifies Deep Neural Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 2498–2507 (cit. on p. 54).

Muralidharan, D. et al. (18, 2019). “Leveraging User Engagement Signals For Entity Labeling in a Virtual Assistant.” In: (cit. on p. 1).

Murphy, K. P. (2012). *Machine learning - A Probabilistic Perspective*. MIT Press (cit. on pp. 17, 18, 40).

Murray, I. and R. P. Adams (2010). “Slice Sampling Covariance Hyperparameters of Latent Gaussian Models.” In: *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., pp. 1732–1740 (cit. on p. 108).

Murray, I. and Z. Ghahramani (2005). *A Note on the Evidence and Bayesian Occam’s Razor*. Tech. rep. GCNU-TR 2005-003. Gatsby Computational Neuroscience Unit, University College London (cit. on pp. 17, 103).

Naeini, M. P., G. F. Cooper, and M. Hauskrecht (2015). “Obtaining Well Calibrated Probabilities Using Bayesian Binning.” In: *AAAI*. AAAI Press, pp. 2901–2907 (cit. on p. 49).

Nalisnick, E. T., J. Gordon, and J. M. Hernández-Lobato (2021). “Predictive Complexity Priors.” In: 130, pp. 694–702 (cit. on p. 84).

Neal, R. M. (1997). “Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification.” In: *Technical Report* (cit. on p. 108).

Neal, R. M. (1992a). “Bayesian Learning via Stochastic Dynamics.” In: *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, pp. 475–482 (cit. on p. 8).

Neal, R. M. (1992b). *Bayesian training of backpropagation networks by the hybrid Monte Carlo method*. Technical Report CRG-TR-92-1. University of Toronto (cit. on p. 8).

Neal, R. M. (1994a). “Bayesian Learning for Neural Networks.” Doctoral dissertation. University of Toronto (cit. on p. 9).

Neal, R. M. (1994b). *Priors for infinite networks*. Technical Report CRG-TR-94-1. University of Toronto (cit. on p. 9).

Neal, R. M. (1996). *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. 1st ed. Springer (cit. on pp. 9, 21, 54, 76, 81, 104).

Neal, R. M. (2011). “MCMC Using Hamiltonian Dynamics.” In: *Handbook of Markov Chain Monte Carlo*. CRC Press. Chap. 5 (cit. on pp. 10, 117).

Newton, M. A. and A. E. Raftery (1994). “Approximate Bayesian inference with the weighted likelihood bootstrap.” English. In: *Journal of the Royal Statistical Society. Series B* 56.1, pp. 3–48 (cit. on p. 11).

Neyshabur, B., Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro (2019). “The role of over-parametrization in generalization of neural networks.” In: *International Conference on Learning Representations* (cit. on p. 76).

Neyshabur, B., R. Tomioka, and N. Srebro (2015). “In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning.” In: *ICLR (Workshop)* (cit. on p. 76).

Nguyen, X., M. J. Wainwright, and M. I. Jordan (2010). “Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization.” In: *IEEE Transactions on Information Theory* 56.11, pp. 5847–5861 (cit. on p. 97).

Niculescu-Mizil, A. and R. Caruana (2005). “Predicting Good Probabilities with Supervised Learning.” In: *Proceedings of the 22nd International Conference on Machine Learning*. ACM, pp. 625–632 (cit. on p. 48).

Nogueira, F. (2014). *Bayesian Optimization: Open source constrained global optimization tool for Python* (cit. on p. 95).

Novak, R., L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz (2020). “Neural Tangents: Fast and Easy Infinite Neural Networks in Python.” In: *Proceedings of the International Conference on Learning Representations* (cit. on p. 135).

Novikov, A., D. Podoprikhin, A. Osokin, and D. P. Vetrov (2015). “Tensorizing Neural Networks.” In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 442–450 (cit. on pp. 55, 62).

O’Hagan, A. (1991). “Bayes–Hermite quadrature.” In: *Journal of Statistical Planning and Inference* 29.3, pp. 245–260 (cit. on p. 79).

Ober, S. W., C. E. Rasmussen, and M. van der Wilk (2021). “The Promises and Pitfalls of Deep Kernel Learning.” In: *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence, UAI 2021, July 27-30, 2021, Virtual Event* (cit. on p. 96).

Oh, K.-S. and K. Jung (2004). “GPU implementation of neural networks.” In: *Pattern Recognition* 37.6, pp. 1311–1314 (cit. on p. 2).

Osawa, K., S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota (2019). “Practical Deep Learning with Bayesian Principles.” In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 4287–4299 (cit. on pp. 9, 79, 104).

Ovadia, Y. et al. (2019). “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift.” In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 13991–14002 (cit. on p. 91).

Pace, F., D. Venzano, D. Carra, and P. Michiardi (2017). “Flexible Scheduling of Distributed Analytic Applications.” In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID ’17)*, pp. 100–109 (cit. on p. 126).

Paszke, A. et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035 (cit. on pp. 3, 26, 43).

Pearce, T., R. Tsuchida, M. Zaki, A. Brintrup, and A. Neely (2019). “Expressive Priors in Bayesian Neural Networks: Kernel Combinations and Periodic Functions.” In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*. AUAI Press, p. 25 (cit. on p. 84).

Pérez-Ortiz, J. A., F. A. Gers, D. Eck, and J. Schmidhuber (2003). “Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets.” In: *Neural Networks* 16, pp. 241–250 (cit. on p. 2).

Pleiss, G. and J. P. Cunningham (2021). “The Limitations of Large Width in Neural Networks: A Deep Gaussian Process Perspective.” In: *CoRR* abs/2106.06529 (cit. on p. 21).

Popper, K. R. (1934). *The Logic of Scientific Discovery*. Hutchinson (cit. on p. 135).

Quiñonero-Candela, J. and C. E. Rasmussen (2005). “A Unifying View of Sparse Approximate Gaussian Process Regression.” In: *Journal of Machine Learning Research* 6.Dec, pp. 1939–1959 (cit. on pp. 110–114).

Rahimi, A. and B. Recht (2008). “Random Features for Large-Scale Kernel Machines.” In: *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., pp. 1177–1184 (cit. on pp. 56, 75, 108).

Ranganath, R., L. Tang, L. Charlin, and D. Blei (2015). “Deep Exponential Families.” In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Vol. 38. PMLR, pp. 762–771 (cit. on p. 34).

Ranzato, M., C. Poultney, S. Chopra, and Y. Leun (2006). “Efficient Learning of Sparse Representations with an Energy-Based Model.” In: *Advances in Neural Information Processing Systems*, pp. 1137–1144 (cit. on p. 2).

Rasmussen, C. E. and Z. Ghahramani (2002). “Bayesian Monte Carlo.” In: *Advances in Neural Information Processing Systems*. Vol. 15. MIT Press, pp. 489–496 (cit. on p. 79).

Rasmussen, C. E. and C. K. I. Williams (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press (cit. on pp. 21, 82, 85, 96, 107, 122).

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). “Stochastic Backpropagation and Approximate Inference in Deep Generative Models.” In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. 2. PMLR, pp. 1278–1286 (cit. on p. 35).

Rezende, D. and S. Mohamed (2015). "Variational Inference with Normalizing Flows." In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. PMLR, pp. 1530–1538 (cit. on pp. 34, 54, 65, 66, 70, 90, 164).

Ritter, H., A. Botev, and D. Barber (2018). "A Scalable Laplace Approximation for Neural Networks." In: *Proceedings of the 6th International Conference on Learning Representations*. OpenReview.net (cit. on p. 10).

Robbins, H. and S. Monro (1951). "A Stochastic Approximation Method." In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407 (cit. on pp. 2, 27).

Roeder, G., Y. Wu, and D. Duvenaud (2017). "Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference." In: *Advances in Neural Information Processing Systems 30*, pp. 6925–6934 (cit. on p. 26).

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, New York (cit. on p. 2).

Rosenblatt, F. (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386 (cit. on p. 2).

Ross, S. M. (2006). *Simulation, Fourth Edition*. Academic Press, Inc. (cit. on p. 25).

Rossi, S., S. Marmin, and M. Filippone (2020). "Walsh-Hadamard Variational Inference for Bayesian Deep Learning." In: *Advances in Neural Information Processing Systems 33* (cit. on pp. 9, 79).

Rossi, S., P. Michiardi, and M. Filippone (2019). "Good Initializations of Variational Bayes for Deep Models." In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, pp. 5487–5497 (cit. on pp. 54, 76, 79).

Ruiz, F. R., M. Titsias, and D. Blei (2016). "The Generalized Reparameterization Gradient." In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc. (cit. on p. 26).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning Internal Representations by Error Propagation." In: *Parallel Distributed Processing*. Vol. 1. MIT Press, pp. 318–362 (cit. on pp. 2, 36).

Salimans, T. and D. A. Knowles (2013). "Fixed-Form Variational Posterior Approximation through Stochastic Linear Regression." In: *Bayesian Analysis* 8.4, pp. 837–882 (cit. on p. 25).

Salimbeni, H. and M. Deisenroth (2017). "Doubly Stochastic Variational Inference for Deep Gaussian Processes." In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 4588–4599 (cit. on p. 120).

Saxe, A. M., J. L. McClelland, and S. Ganguli (2013). "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks." In: *CoRR* abs/1312.6, pp. 1–22 (cit. on pp. 35, 37, 43).

Schaul, T., S. Zhang, and Y. LeCun (2013). “No more pesky learning rates.” In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. 3. PMLR, pp. 343–351 (cit. on p. 2).

Schmidhuber, J. (2015). “Deep learning in neural networks: An overview.” In: *Neural Networks* 61, pp. 85–117 (cit. on p. 2).

Schmidt, R. M., F. Schneider, and P. Hennig (2021). “Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers.” In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, pp. 9367–9376 (cit. on p. 2).

Schuld, M. and F. Petruccione (2021). *Machine learning with quantum computers*. English. Springer, pp. xiv + 312 (cit. on p. 135).

Seeger, M., C. K. Williams, and N. D. Lawrence (2003). “Fast Forward Selection to Speed Up Sparse Gaussian Process Regression.” In: *Artificial Intelligence and Statistics* (cit. on p. 113).

Sejdinovic, D., H. Strathmann, M. L. Garcia, C. Andrieu, and A. Gretton (2014). “Kernel Adaptive Metropolis-Hastings.” In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. 2. PMLR, pp. 1665–1673 (cit. on p. 56).

Shi, J., M. K. Titsias, and A. Mnih (2020). “Sparse Orthogonal Variational Inference for Gaussian Processes.” In: *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*. Vol. 108. PMLR, pp. 1932–1942 (cit. on pp. 108, 109).

Silva, F. M. and L. B. Almeida (1990). “Speeding Up Back-Propagation.” In: *Advanced Neural Computers*. Elsevier, pp. 151–158 (cit. on p. 2).

Simard, P. and Y. LeCun (1993). “Local Computation of the Second Derivative Information in a Multi-Layer Network.” In: *Advances in Neural Information Processing Systems 5*. CA: Morgan Kaufmann (cit. on p. 2).

Simonyan, K. and A. Zisserman (2014). “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: *CoRR* abs/1409.1556 (cit. on pp. 47, 49, 70, 93).

Snelson, E. L. (2007). “Flexible and efficient Gaussian process models for machine learning.” Doctoral dissertation. UCL (University College London) (cit. on p. 114).

Snelson, E. and Z. Ghahramani (2006). “Sparse Gaussian Processes using Pseudo-Inputs.” In: *Advances in Neural Information Processing Systems 18*. MIT Press, pp. 1257–1264 (cit. on pp. 111, 112).

Snoek, J., H. Larochelle, and R. P. Adams (2012). “Practical Bayesian Optimization of Machine Learning Algorithms.” In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc. (cit. on p. 95).

Sønderby, C. K., T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther (2016). "Ladder Variational Autoencoders." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 3738–3746 (cit. on pp. 50, 54, 76).

Springenberg, J. T., A. Klein, S. Falkner, and F. Hutter (2016). "Bayesian Optimization with Robust Bayesian Neural Networks." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 4134–4142 (cit. on pp. 10, 31, 67, 82, 90, 91).

Srinivas, N., A. Krause, S. M. Kakade, and M. W. Seeger (2010). "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design." In: *Proceedings of the 27th International Conference on Machine Learning*. Omnipress, pp. 1015–1022 (cit. on p. 79).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958 (cit. on pp. 33, 35).

Strathmann, H., D. Sejdinovic, S. Livingstone, Z. Szabo, and A. Gretton (2015). "Gradient-free Hamiltonian Monte Carlo with Efficient Kernel Exponential Families." In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 955–963 (cit. on p. 56).

Strauss, D. J. (1975). "A Model for Clustering." In: *Biometrika* 62.2, pp. 467–475 (cit. on p. 118).

Su, Y., D. Vandyke, S. Wang, Y. Fang, and N. Collier (2021). "Plan-then-Generate: Controlled Data-to-Text Generation via Planning." In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 895–909 (cit. on p. 1).

Sun, S., C. Chen, and L. Carin (2017). "Learning Structured Weight Uncertainty in Bayesian Neural Networks." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Vol. 54. PMLR, pp. 1283–1292 (cit. on p. 9).

Sun, S., G. Zhang, J. Shi, and R. B. Grosse (2019). "Functional Variational Bayesian Neural Networks." In: *Proceedings of the 7th International Conference on Learning Representations*. OpenReview.net (cit. on pp. 10, 84).

Sutskever, I., J. Martens, G. Dahl, and G. Hinton (2013). "On the importance of initialization and momentum in deep learning." In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. 3. PMLR, pp. 1139–1147 (cit. on p. 37).

Thomas Bayes (1763). "An essay towards solving a problem in the doctrine of chances." In: *Philosophical Transactions of the Royal Society of London* 53, pp. 370–418 (cit. on p. 15).

Tieleman, T. and G. Hinton (2012). *RMSProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning (cit. on p. 31).

Tishby, Levin, and Solla (1989). “Consistent inference of probabilities in layered networks: predictions and generalizations.” In: *International 1989 Joint Conference on Neural Networks*, 403–409 vol.2 (cit. on p. 8).

Titsias, M. K. (2009a). “Variational Model Selection for Sparse Gaussian Process Regression.” In: *Report, University of Manchester, UK* (cit. on pp. 113, 116, 124).

Titsias, M. K. (2009b). “Variational Learning of Inducing Variables in Sparse Gaussian Processes.” In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009*. Vol. 5. JMLR.org, pp. 567–574 (cit. on pp. 74, 108, 109, 111–114, 123).

Tran, B., S. Rossi, D. Milios, P. Michiardi, E. V. Bonilla, and M. Filippone (2021). “Model Selection for Bayesian Autoencoders.” In: *CoRR* abs/2106.06245 (cit. on p. 103).

Tran, G.-L., E. V. Bonilla, J. Cunningham, P. Michiardi, and M. Filippone (2019). “Calibrating Deep Convolutional Gaussian Processes.” In: *Proceedings of Machine Learning Research*. Vol. 89. PMLR, pp. 1554–1563 (cit. on p. 76).

Tran, B.-H., S. Rossi, D. Milios, and M. Filippone (2020). *All You Need is a Good Functional Prior for Bayesian Deep Learning* (cit. on p. 94).

Tropp, J. A. (2011). “Improved Analysis of the subsampled Randomized Hadamard Transform.” In: *Advances in Adaptive Data Analysis* 3.1-2, pp. 115–126 (cit. on pp. 55, 57).

Van den Berg, R., L. Hasenclever, J. M. Tomczak, and M. Welling (2018). “Sylvester Normalizing Flows for Variational Inference.” In: *UAI ’18: Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence* (cit. on pp. 66, 70, 164).

Villani, C. (2003). *Topics in Optimal Transportation*. American Mathematical Society (cit. on pp. 86, 167).

Villani, C. (2008). *Optimal Transport: Old and New*. Vol. 338. Springer Science & Business Media (cit. on p. 98).

Vogl, T., J. Mangis, A. Rigler, W. Zink, and D. Alkon (1988). “Accelerating the Convergence of the Back-Propagation Method.” In: *Biological Cybernetics* 59, pp. 257–263 (cit. on p. 2).

Wallach, H. M., D. M. Mimno, and A. McCallum (2009). “Rethinking LDA: Why Priors Matter.” In: *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., pp. 1973–1981 (cit. on p. 130).

Wan, N., D. Li, and N. Hovakimyan (2020). “f-Divergence Variational Inference.” In: *Advances in Neural Information Processing Systems 33* (cit. on p. 23).

Wang, J. X., Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick (2018). “Prefrontal cortex as a meta-reinforcement learning system.” In: *Nature Neuroscience* 21.6, pp. 860–868 (cit. on p. 3).

Wen, W., C. Wu, Y. Wang, Y. Chen, and H. Li (2016). "Learning Structured Sparsity in Deep Neural Networks." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 2074–2082 (cit. on p. 71).

Wenzel, F. et al. (2020). "How Good is the Bayes Posterior in Deep Neural Networks Really?" In: *Proceeding of the 37th International Conference on Machine Learning* (cit. on pp. 54, 76, 80, 82, 91, 94).

Wilcoxon, F. (1945). "Individual Comparisons by Ranking Methods." In: *Biometrics Bulletin* 1.6, pp. 80–83 (cit. on p. 124).

Wilk, M. van der, C. E. Rasmussen, and J. Hensman (2017). "Convolutional Gaussian Processes." In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 2849–2858 (cit. on p. 108).

Wilson, A. G., Z. Hu, R. R. Salakhutdinov, and E. P. Xing (2016). "Stochastic Variational Deep Kernel Learning." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 2586–2594 (cit. on p. 108).

Wilson, A. G., Z. Hu, R. Salakhutdinov, and E. P. Xing (2016). "Deep Kernel Learning." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*. Vol. 51. PMLR, pp. 370–378 (cit. on pp. 127, 128).

Wilson, A. G. and P. Izmailov (2020a). *Bayesian Deep Learning and a Probabilistic Perspective of Generalization* (cit. on pp. 54, 76, 80).

Wilson, A. G. and P. Izmailov (2020b). "Bayesian Deep Learning and a Probabilistic Perspective of Generalization." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (cit. on p. 103).

Wilson, A. and H. Nickisch (2015). "Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)." In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1775–1784 (cit. on pp. 108, 127).

Wilt, J. K., C. Yang, and G. X. Gu (2020). "Accelerating Auxetic Metamaterial Design with Deep Learning." In: *Advanced Engineering Materials* 22.5, p. 1901266 (cit. on p. 3).

Wu, L. (2021). "How to leverage the multimodal EHR data for better medical prediction?" In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Microsoft (cit. on p. 1).

Yamins, D. L. and J. J. DiCarlo (2016). "Using goal-driven deep learning models to understand sensory cortex." In: *Nature Neuroscience*. Vol. 19. 3, pp. 356–365 (cit. on p. 3).

Yang, G. (2019). "Wide Feedforward or Recurrent Neural Networks of Any Architecture are Gaussian Processes." In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc. (cit. on p. 21).

Yang, W., L. Lorch, M. A. Graule, S. Srinivasan, A. Suresh, J. Yao, M. F. Pradier, and F. Doshi-velez (2019). "Output-Constrained Bayesian Neural Networks." In: *ICML workshop on Uncertainty & Robustness in Deep Learning* (cit. on p. 84).

Yang, Z., M. Moczulski, M. Denil, N. d. Freitas, A. Smola, L. Song, and Z. Wang (2015). "Deep Fried Convnets." In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1476–1483 (cit. on p. 76).

Yu, F. X., A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar (2016). "Orthogonal Random Features." In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pp. 1975–1983 (cit. on pp. 55, 57, 76).

Yu, H., Y. Chen, B. K. H. Low, P. Jaillet, and Z. Dai (2019). "Implicit Posterior Variational Inference for Deep Gaussian Processes." In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 14475–14486 (cit. on pp. 108, 122, 125).

Yu, X.-H., G.-A. Chen, and S.-X. Cheng (1995). "Dynamic Learning Rate Optimization of the Backpropagation Algorithm." In: *IEEE Transactions on Neural Networks* 6.3, pp. 669–677 (cit. on p. 2).

Yurtsever, E., J. Lambert, A. Carballo, and K. Takeda (2020). "A Survey of Autonomous Driving: Common Practices and Emerging Technologies." In: *IEEE Access* 8, pp. 58443–58469 (cit. on p. 1).

Zador, A. M. (2019). "A critique of pure learning and what artificial neural networks can learn from animal brains." In: *Nature Communications*. Vol. 10. 1 (cit. on p. 3).

Zaremba, W., A. Gretton, and M. Blaschko (2013). "B-test: A Non-parametric, Low Variance Kernel Two-sample Test." In: *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., pp. 755–763 (cit. on p. 56).

Zeiler, M. D. (2012). "ADADELTA: An Adaptive Learning Rate Method." In: *CoRR* abs/1212.5701 (cit. on pp. 2, 37).

Zhang, F., B. Shao, G. Xu, B. Yang, Z. Yang, Z. Qin, K. Ren, and Z. Yang (2020). "From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices." In: *Proceedings - Design Automation Conference* (cit. on p. 3).

Zhang, G., S. Sun, D. Duvenaud, and R. Grosse (2018). "Noisy Natural Gradient as Variational Inference." In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, pp. 5852–5861 (cit. on pp. 9, 35, 44, 49, 50, 55, 68).

Zhang, R., C. Li, J. Zhang, C. Chen, and A. G. Wilson (2020). "Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning." In: *Proceedings of the 8th International Conference on Learning Representations*. OpenReview.net (cit. on pp. 10, 104).

Zhang, Z. and G. X. Gu (2020). "Finite-Element-Based Deep-Learning Model for Deformation Behavior of Digital Materials." In: *Advanced Theory and Simulations* 3.7, p. 2000031 (cit. on p. 3).

Zhao, Z., J. K. Fitzsimons, and J. F. Fitzsimons (2019). "Quantum-assisted Gaussian process regression." In: *Physical Review A* 99 (5), p. 052331 (cit. on p. 135).

Zhizhou Zhang, G. X. G. (2021). "Physics-informed deep learning for digital materials." In: *Theoretical & Applied Mechanics Letters* 11, p. 1 (cit. on p. 3).

Zhu, M. and S. Gupta (2018). "To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression." In: *ICLR (Workshop)*. OpenReview.net (cit. on p. 76).

Zhu, M. and A. Y. Lu (2004). "The Counter-intuitive Non-informative Prior for the Bernoulli Family." In: *Journal of Statistics Education* 12.2 (cit. on p. 16).

a

ADDITIONAL DERIVATIONS

A.1 RECENT ADVANCES ON BAYESIAN INFERENCE FOR DEEP MODELS

Consider an invertible, continuous and differentiable function $f : \mathbb{R}^D \to \mathbb{R}^D$. Given $\tilde{z}_0 \sim q(z_0)$, then $\tilde{z}_1 = f(\tilde{z}_0)$ follows $q(z_1)$ defined as

$$q(z_1) = q(z_0) \left| \det \frac{\partial f}{\partial z_0} \right|^{-1} . \tag{a.1}$$

As a consequence, after K transformations the log-density of the final distribution is

$$\log q(z_K) = \log q(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_{k-1}}{\partial z_{k-1}} \right| . \tag{a.2}$$

We shall define $f_k(z_{k-1}; \lambda_k)$ the k^{th} transformation which takes input from the previous flow z_{k-1} and has parameters λ_k. The final variational objective is

$$\begin{aligned}
\mathcal{L}(\theta, \phi) &= -\mathbb{E}_{q_\phi(z)}[\log p_\theta(x \mid z)] + \mathrm{KL}\left[q_\phi(z) \parallel p(z)\right] = \\
&= \mathbb{E}_{q_\phi(z)}[-\log p_\theta(x \mid z) - \log p(z) + \log q_\phi(z)] = \\
&= \mathbb{E}_{q_0(z_0)}[-\log p_\theta(x \mid z_K) - \log p(z_K) + \log q_K(z_K)] = \\
&= \mathbb{E}_{q_0(z_0)}\left[-\log p_\theta(x \mid z_K) - \log p(z_K) + \log q_0(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k(z_{k-1}; \lambda_k)}{\partial z_{k-1}} \right| \right] = \\
&= -\mathbb{E}_{q_0(z_0)} \log p_\theta(x \mid z) + \mathrm{KL}\left[q_0(z_0) \parallel p(z_K)\right] - \mathbb{E}_{q_0(z_0)} \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k(z_{k-1}; \lambda_k)}{\partial z_{k-1}} \right| .
\end{aligned} \tag{a.3}$$

Setting the initial distribution q_0 to a fully factorized Gaussian $\mathcal{N}(\mathbf{z}_0 \,|\, \boldsymbol{\mu}, \boldsymbol{\sigma}\mathbf{I})$ and assuming a Gaussian prior on the generated $\mathbf{z}_K$, the Kullback-Leibler (KL) term is analytically tractable. A possible family of transformation is the *planar flow* (D. Rezende and Mohamed, 2015). For the *planar flow*, f is defined as

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\top \mathbf{z} + b)\,, \tag{a.4}$$

where $\lambda = [\mathbf{u} \in \mathbb{R}^D, \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}]$ and $h(\cdot) = \tanh(\cdot)$. This is equivalent to a residual layer with single neuron MLP – as argued by Kingma, Salimans, et al. (2016). The log-determinant of the Jacobian of f is

$$\begin{aligned} \log\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| &= \left|\det\Big(\mathbf{I} + \mathbf{u}[h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w}]^\top\Big)\right| \\ &= \left|1 + \mathbf{u}^\top \mathbf{w} h'(\mathbf{w}^\top \mathbf{z} + b)\right| \,. \end{aligned} \tag{a.5}$$

Alternatives can be found in recent works by D. Rezende and Mohamed (2015), Van den Berg et al. (2018), Kingma, Salimans, et al. (2016), and Louizos and Welling (2017).

A.2 A PRIMER OF WASSERSTEIN DISTANCE

Given two Borel's probability measures $\pi(\mathbf{x})$ and $\nu(\mathbf{y})$ defined on the Polish space $\mathcal{X}$ and $\mathcal{Y}$ (i.e. any complete separable metric space such as a subset of $\mathbb{R}^d$), the p-Wasserstein distance is defined as follows

$$W_p(\pi, \nu) = \left(\inf_{\gamma \in \Gamma(\pi,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} D(\mathbf{x}, \mathbf{y})^p \gamma(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\, d\mathbf{y}\right)^{1/p}, \tag{a.6}$$

where $D(\mathbf{x}, \mathbf{y})$ is a proper distance metric between two points $\mathbf{x}$ and $\mathbf{y}$ in the space $\mathcal{X} \times \mathcal{Y}$ and $\Gamma(\pi, \nu)$ is the set of functionals of all possible joint densities whose marginals are indeed π and ν.

When the space of $\mathbf{x}$ and $\mathbf{y}$ coincides (i.e. $\mathbf{x}, \mathbf{y} \in \mathcal{X} \subseteq \mathbb{R}^d$), the most used formulation is the 1-Wasserstein distance with Euclidian norm as distance,

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi,\nu)} \int_{\mathcal{X}\times\mathcal{X}} \|\mathbf{x} - \mathbf{y}\| \gamma(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\, d\mathbf{y}\,, \tag{a.7}$$

This is also known in the literature as the Earth-Mover distance. Intuitively, here γ measures how much mass must be transported from $\mathbf{x}$ to $\mathbf{y}$ in order to transform the distributions π into the distribution ν. Solving the Wasserstein distance means computing the minimum mass that needs to be moved. The question "How?" is answered by looking at the optimal transport plan (not the focus of these notes).

The remaining part of these notes will be dedicated to the proof of the dual formulation for Equation (a.7). It is well known in the literature of optimization that linear programming problem with convex constrains admits a dual formulation. Kantorovich introduced the dual formulation of the Wasserstein distance in 1942.

Theorem 1 *On the same setup as before, the Wasserstein distance defined as*

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\| \gamma(x, y) \, dx \, dy \,, \tag{a.8}$$

admits the following dual form

$$W(\pi, \nu) = \sup_{\|f\|_L \leqslant 1} \int_{\mathcal{X}} f(x) \pi(x) dx - \int_{\mathcal{X}} f(y) \nu(y) dy \tag{a.9}$$

where f *is a 1-Lipschitz continuous function defined on* $\mathcal{X} \to \mathbb{R}$.

Step 1: Kantorovich duality

First of all we start with the **Kantorovich duality**, which defines a dual form for the generic 1-Wasserstein.

Theorem 2 *Given a nonnegative measurable function* $D : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*, the 1-Wasserstein is computed as follows,*

$$W(\pi, \nu) = \inf_{\gamma \in \Gamma(\pi, \nu)} \int D(x, y) \gamma(x, y) \, dx \, dy \,, \tag{a.10}$$

The Kantorovich duality proves that this is equal to the following constrained optimization problem,

$$W(\pi, \nu) = \sup_{\substack{f, g \\ f(x) + g(y) \leqslant D(x, y)}} \int f(x) \pi(x) \, dx + \int g(y) \nu(y) \, dy \,. \tag{a.11}$$

We define $\iota_\Gamma(\gamma)$ the following quantity

$$\iota_\Gamma(\gamma) = \sup_{f,g} \left[\int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy - \iint [f(x) + g(y)]\,\gamma(x,y)\,dx\,dy \right] \tag{a.12}$$

and we observe that

$$\iota_\Gamma(\gamma) = \begin{cases} 0 & \text{if } \gamma \in \Gamma(\pi,\nu)\,, \\ +\infty & \text{otherwise}\,. \end{cases} \tag{a.13}$$

This is true because given the definition of Γ, if $\gamma \in \Gamma(\pi,\nu)$ then $\pi(x) = \int \gamma(x,y)\,dy$ and $\nu(y) = \int \gamma(x,y)\,dx$. By substituiting these quantities, it follows that

$$\begin{aligned} \int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy &= \int f(x) \int \gamma(x,y)\,dy\,dx + \int g(y) \int \gamma(x,y)\,dx\,dy \\ &= \iint [f(x) + g(y)]\,\gamma(x,y)\,dx\,dy\,. \end{aligned} \tag{a.14}$$

In other cases, f and g can be chosen such that the supremum becomes $+\infty$. Given this property and the constrain on γ, we can add $\iota_\Gamma(\gamma)$ to the formulation of the Wasserstein distance in Equation (a.8),

$$\begin{aligned} W(\pi,\nu) = &\inf_{\gamma\in\Gamma(\pi,\nu)} \left[\iint D(x,y)\gamma(x,y)\,dx\,dy \right] + \iota_\Gamma(\gamma) = \\ &= \inf_{\gamma} \left[\int D(x,y)\gamma(x,y)\,dx\,dy + \sup_{f,g} \left[\int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy - \right.\right. \\ &\qquad \left.\left. \iint [f(x) + g(y)]\,\gamma(x,y)\,dx\,dy \right] \right]\,, \end{aligned} \tag{a.15}$$

Now, the original integral of the Wasserstein distance does not depend on f and g; therefore the supremum can be moved in front,

$$\begin{aligned} W(\pi,\nu) &= \inf_{\gamma} \sup_{f,g} \Upsilon(\gamma,(f,g)) \\ \Upsilon(\gamma,(f,g)) &\overset{\text{def}}{=} \int D(x,y)\gamma(x,y)\,dx\,dy + \int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy - \\ &\qquad \iint [f(x) + g(y)]\,\gamma(x,y)\,dx\,dy \end{aligned} \tag{a.16}$$

Under certain conditions stated by the *minimax theorem*, i.e. $\Upsilon(\gamma,(f,g))$ is convex-concave function (Υ is concave for fixed (f,g) while convex for fixed γ), we can swap the infinum and the supremum and rewrite the definition as follows,

$$W(\pi,\nu) = \sup_{f,g} \inf_{\gamma} \int [D(x,y) - f(x) - g(y)]\,\gamma(x,y)\,dx\,dy + \int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy \tag{a.17}$$

Proofs that the hypothesis used for the minimax theorem hold for this case are presented in Theorem 1.9 of "Topics in Optimal Transport" (Villani, 2003). Focusing on the infimum part, we can write

$$\inf_{\gamma} \int [D(x,y) - f(x) - g(y)]\,\gamma(x,y)\,dx\,dy = \begin{cases} 0 & \text{if } f(x) + g(y) \leqslant D(x,y)\,, \\ -\infty & \text{otherwise}\,. \end{cases} \tag{a.18}$$

If the function $\zeta(x,y) = D(x,y) - (f(x) + g(y))$ takes a negative value at some point (x_0,y_0), then by choosing $\gamma = \lambda\delta(x_0,y_0)$ with $\lambda \to +\infty$ (i.e. a Dirac delta in (x_0,y_0)), we see that the infimum is infinite. On the other hand, is $\zeta(x,y)$ is nonnegative, then the infimum is obtained for $\gamma = 0$. Finally, this constrains can be added to the previous conditions making thus recovering the formulation in Equation (a.9).

Step 2: D-Transforms

The next challenge is to find f and g such that we can easily recover the constrain optimization above. We approach this problem by supposing to have chosen some $f(x)$. This means that the objective is to find a good $g(y)$ that for all x,y satisfy the condition

$$f(x) + g(y) \leqslant D(x,y)\,. \tag{a.19}$$

The trivial solution is $g(y) \leqslant D(x,y) - f(x)$. This must be true for all x, also in the worst case (when we take the infimum),

$$g(y) \leqslant \inf_{x} [D(x,y) - f(x)]\,. \tag{a.20}$$

At this point, we observe that for a given f, if we want the supremum in Eq. 5 we cannot get a better g then taking the equality,

$$\bar{f}(y) := \inf_{x} \left[D(x, y) - f(x) \right]. \tag{a.21}$$

We therefore have the following formulation of the Wasserstein distance,

$$W(\pi, \nu) = \sup_{f} \left[\int f(x) \pi(x)\, dx + \int \bar{f}(y) \nu(y)\, dy \right] \tag{a.22}$$

If now we suppose to choose g, by following the same reasoning the best f that we can get is defined

$$\bar{\bar{f}}(x) = \bar{g}(x) := \inf_{y} \left[D(x, y) - g(y) \right]. \tag{a.23}$$

If we replace $g(y)$ with Eq. 17 we have yet another recursive definition of the Wasserstein distance,

$$W(\pi, \nu) = \sup_{f} \left[\int \bar{\bar{f}}(x) \pi(x)\, dx + \int \bar{f}(y) \nu(y)\, dy \right] \tag{a.24}$$

If we constrain f to be D-concave, then $\bar{\bar{f}} = f$.

Step 2.1: Euclidean distance

It's worth mentioning that this formulation is valid for any nonnegative measurable function D. For the Euclidian distance this simplify even further.

Theorem 3 *When* $D(x, y) = \|x - y\|$ *and* f *is 1-Lipschitz,* f *is* D*-concave if and only if* $\bar{f} = -f$

We prove the necessity condition of such result. First of all, we observe that if f is 1-Lipschitz then $\bar{f}$ is 1-Lipschitz too. This is true because for any given x

$$\bar{f}_x(y) = \|x - y\| - f(x) \tag{a.25}$$

is 1-Lipschitz and therefore the infimum of $\bar{f}(y) = \inf_x \|x - y\| - f(x)$ is 1-Lipschitz. Since $\bar{f}$ is 1-Lipschitz, for all x and y we have

$$\left| \bar{f}(y) - \bar{f}(x) \right| \leqslant \|y - x\| \implies -\bar{f}(x) \leqslant \|x - y\| - \bar{f}(y) \tag{a.26}$$

Since this is true for all y,

$$\begin{aligned}
-\bar{f}(x) &\leqslant \inf_{y} \|x-y\| - \bar{f}(y) \\
-\bar{f}(x) &\leqslant \underbrace{\inf_{y} \|x-y\| - \bar{f}(y)}_{\bar{\bar{f}} \equiv f} \leqslant -\bar{f}(x)
\end{aligned} \tag{a.27}$$

where the right inequality follows by choosing $y = x$ in the infimum. We know that $\bar{\bar{f}} \equiv f$. This means that $-\bar{f}(x)$ must be equal to $f(x)$ for the last equation to hold.

Step 3. Putting everything together

We started our discussion by proving the Kantovich duality, which states that

$$\inf_{\gamma \in \Gamma(\pi,\nu)} \int D(x,y)\gamma(x,y)\,dx\,dy = \sup_{\substack{f,g \\ f(x)+g(y) \leqslant D(x,y)}} \int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy\,, \tag{a.28}$$

We then proved that

$$\begin{aligned}
&\sup_{\substack{f,g \\ f(x)+g(y) \leqslant D(x,y)}} \left[\int f(x)\pi(x)\,dx + \int g(y)\nu(y)\,dy\right] = \\
&\quad = \sup_{\substack{f \\ \bar{f} = \inf_x D - f}} \left[\int f(x)\pi(x)\,dx + \int \bar{f}(y)\nu(y)\,dy\right],
\end{aligned}$$

Finally, given $D(x,y)$ to be the Euclidean distance, we discussed the shape of $\bar{f}$ when we restrict f to be 1-Lipschitz, showing that $\bar{f} = -f$. Putting everything together, we obtain the dual 1-Wasserstein distance in Equation (a.9),

$$W(\pi,\nu) = \sup_{\|f\|_L \leqslant 1} \int f(x)\pi(x)dx - \int f(y)\nu(y)dy \tag{a.29}$$

b

ADDITIONAL MATERIAL FOR CHAPTER 3

B.1 EXPERIMENTS

REGRESSION ON SHALLOW NETWORKS In this experiment we compare initialization methods for a shallow deep neural network (DNN) architecture on two datasets. The architecture used in these experiments has one single hidden layer with 100 hidden neurons and rectified linear unit (ReLU) activations. We impose that the approximate posterior has fully factorized covariance. Figure b.1 shows the learning curves on the Powerplant ($n = 9568$, $d = 4$) and Protein ($n = 45730$, $d = 9$) datasets, repeated over five different train/test splits. iterative Bayesian linear modeling (IBLM) allows for a better initialization compared to the competitors, leading to a lower root mean square error (RMSE) and lower mean negative log-likelihood (mean negative loglikelihood (MNLL)) on the test for a given computational budget. We refer the reader to the supplementary material for a more detailed analysis of the results.

CLASSIFICATION WITH A DEEP ARCHITECTURE Using the same deep DNN architecture as in the last experiment (five hidden layers with 100 neurons), we tested IBLM with classification problems on MNIST ($n = 70000$, $d = 784$), EEG ($n = 14980$, $d = 14$), Credit ($n = 1000$, $d = 24$) and Spam ($n = 4601$, $d = 57$). Interestingly, with this architecture, some initialization strategies struggled to converge, e.g., uninformative on MNIST and LSUV on EEG . The gains offered by IBLM achieves are most apparent on MNIST . After less than 1000 training steps (less than an epoch), IBLM makes variational inference (VI) reach a test accuracy greater than 95%; other initializations reach such performance much later during training. Even

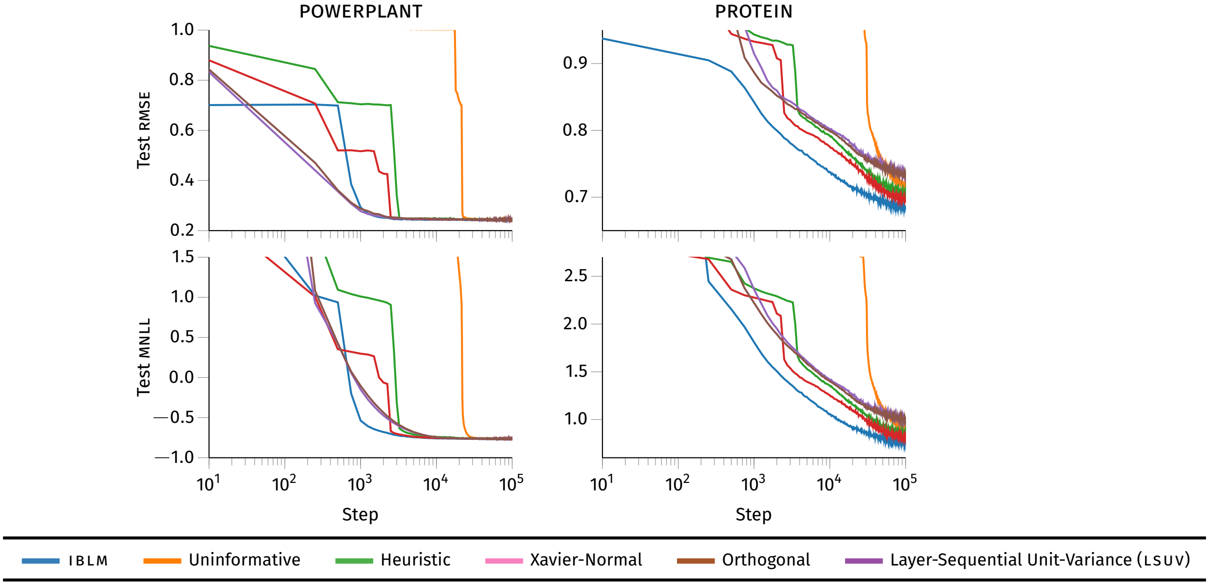


Figure b.1: Progression of test RMSEand test MNLL with different initializations on a shallow architecture.

after 100 epochs, VI inference initialized with IBLM provides on average an increase up to 14% of accuracy at test time. Full results are reported in the supplementary material.

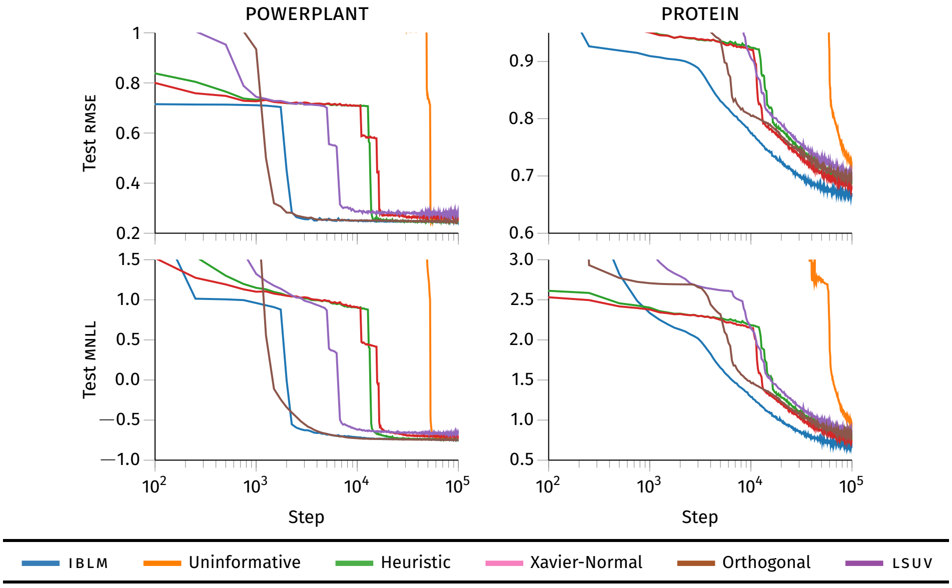


Figure b.2: Progression of test RMSEand test MNLL with different initializations on a deep architecture.

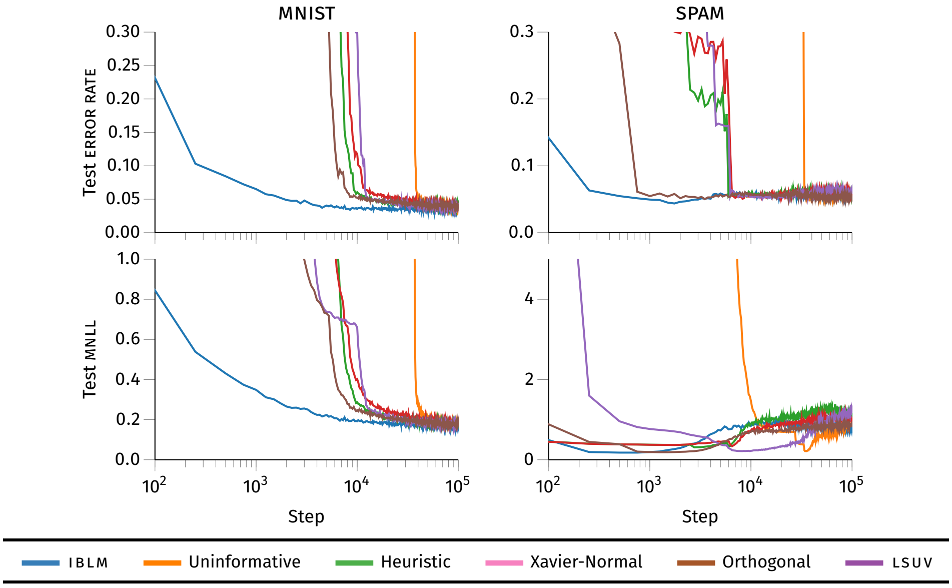


Figure b.3: Progression of test RMSEand test MNLL with different initializations on a deep architecture.

COLOPHON

This document was typeset with LaTeX using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić.

Final Version as of December 17, 2021 (1).