Nonparametric Bayesian Deep Learning for Scientific Data Analysis

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Devanshu Agrawal

December 2020

Copyright © by Devanshu Agrawal, 2020
All Rights Reserved.

Acknowledgments

There have been many people over the years who have been instrumental in making this dissertation come to fruition. First and foremost, I would like to thank my advisor Dr. Jacob Hinkle, who has shown me immense patience and has encouraged me the whole way, but who has also challenged me to think critically. We tossed around a lot of wild ideas during our many long conversations, some of which made it into this dissertation. I would also like to thank my collaborators Dr. Abhishek Dubey and Dr. Theodore Papamarkou for their support and time. I would like to thank Dr. Georgia Tourassi for giving me the opportunity to work in her research group at Oak Ridge National Laboratory. I would like to thank Jacki Isaacs for holding down the fort in the group and the entire group for accepting me. I would like to thank the rest of my committee, Dr. Vasileios Maroulas and Dr. Hairong Qi, not just for their input into my dissertation but also for the amazing courses they taught. I would like to thank Dr. Russell Zaretzki, Dr. Lee Riedinger, Dr. Suresh Babu, Wanda Davis, Allie Burns and everyone else at the Bredesen Center for welcoming me into their "family" and letting me start this journey. I would like to thank the many instructors I have had in the past– Dr. Jeff Knisley for sparking my interest in machine learning as an undergraduate, Mr. Guy Mauldin for making me realize that I love mathematics in high school, and Ms. Nila Collins who helped me persevere through the hardest time of my life. I would like to thank my friends Shang Gao, Will Halsey, John Lagergren, and Ty Frazier for their support, advice, crazy conversations, and even their jokes. Finally and certainly not the least, I would like to thank my parents Dr. Alok Agrawal and Vandana Agrawal for believing in me even when I didn't believe in myself, and my sister Shefali Agrawal, who listened to me go on about probability and algorithms even when she didn't know what I was talking about.

Abstract

Deep learning (DL) has emerged as the leading paradigm for predictive modeling in a variety of domains, especially those involving large volumes of high-dimensional spatio-temporal data such as images and text. With the rise of big data in scientific and engineering problems, there is now considerable interest in the research and development of DL for scientific applications. The scientific domain, however, poses unique challenges for DL, including special emphasis on interpretability and robustness. In particular, a priority of the Department of Energy (DOE) is the research and development of probabilistic ML methods that are robust to overfitting and offer reliable uncertainty quantification (UQ) on high-dimensional noisy data that is limited in size relative to its complexity.

Gaussian processes (GPs) are nonparametric Bayesian models that are naturally robust to overfitting and offer UQ out-of-the-box. Unfortunately, traditional GP methods lack the balance of expressivity and domain-specific inductive bias that is key to the success of DL. Recently, however, a number of approaches have emerged to incorporate the DL paradigm into GP methods, including deep kernel learning (DKL), deep Gaussian processes (DGPs), and neural network Gaussian processes (NNGPs).

In this work, we investigate DKL, DGPs, and NNGPs as paradigms for developing robust models for scientific applications. First, we develop DKL for text classification, and apply both DKL and Bayesian neural networks (BNNs) to the problem of classifying cancer pathology reports, with BNNs attaining new state-of-the-art results. Next, we introduce the deep ensemble kernel learning (DEKL) method, which is just as powerful as DKL while admitting easier model parallelism. Finally, we derive a new model called a "bottleneck NNGP" by unifying the DGP and NNGP paradigms, thus laying the groundwork for a new class of methods for future applications.

Table of Contents

1 Introduction **1**

2 Background **7**
2.1 Bayesian Neural Networks . . . 8
2.2 Gaussian Processes . . . 9
2.3 Deep Kernel Learning . . . 11
2.4 Deep Gaussian Processes . . . 12
2.5 Neural Network Gaussian Processes . . . 13
2.6 Information Extraction from Cancer Pathology Reports . . . 14

3 Deep Kernel Learning for Medical and Non-Medical Text Classification Tasks **17**
3.1 Introduction . . . 18
3.2 Datasets . . . 20
3.2.1 Electronic Pathology Reports . . . 20
3.2.2 Public Text Datasets . . . 21
3.3 Classification methods . . . 21
3.3.1 Convolutional neural networks . . . 21
3.3.2 Gaussian process classifiers . . . 25
3.3.3 Deep kernel learning . . . 27
3.3.4 Uncertainty quantification . . . 28
3.4 Experimental setup . . . 28
3.4.1 Metrics . . . 29
3.4.2 Models . . . 29
3.4.3 Training details . . . 31
3.5 Results . . . 31
3.5.1 Model performance . . . 31
3.5.2 Feature visualizations . . . 32
3.5.3 Comparison of uncertainty quantification . . . 37
3.6 Discussion . . . 37

4 Bayesian Neural Networks for Information Extraction from Cancer Pathology Reports **41**
4.1 Introduction . . . 42
4.2 Materials and methods . . . 44
4.2.1 Dataset . . . 44

4.2.2 Models . . . 44
4.2.3 Experimental setup . . . 47
4.3 Results . . . 48
4.4 Discussion . . . 52

5 Deep Ensemble Kernel Learning **53**
5.1 Introduction . . . 54
5.2 Deep ensemble kernel learning . . . 56
5.2.1 Exact inference . . . 58
5.2.2 Variational inference . . . 59
5.2.3 Optimal prior covariance . . . 60
5.3 Experiments . . . 61
5.3.1 Synthetic data . . . 61
5.3.2 Real-World data . . . 63
5.3.3 Diversity of learners . . . 63
5.4 Discussion . . . 66

6 Wide Neural Networks with Bottlenecks are Deep Gaussian Processes **69**
6.1 Introduction . . . 70
6.2 Preliminaries . . . 73
6.2.1 Deep Gaussian processes . . . 73
6.2.2 Wide neural networks as GPs . . . 74
6.2.3 The no-bottleneck NNGP limit . . . 76
6.3 The bottleneck NNGP theorem . . . 78
6.3.1 Statement of main theorem . . . 78
6.3.2 Correspondence to the no-bottleneck NNGP . . . 79
6.3.3 Experiments . . . 80
6.4 Bottleneck layers induce dependence . . . 84
6.4.1 An exact formula for quadratic correlation . . . 84
6.4.2 Quadratic correlation as a function of depth . . . 89
6.4.3 A divergent depth scale . . . 93
6.4.4 Non-degenerate kernels at extreme depths . . . 93
6.4.5 Other nonlinearities . . . 97
6.4.6 Other deep limits . . . 99
6.5 Discussion . . . 99

7 Conclusion **101**

Bibliography **104**

Appendices **118**

Vita **165**

List of Tables

Table 3.1: Number of classes, average number of training points per class, and average number of test points per class for each dataset. 22
Table 3.2: Classifier, pretraining, and feature network specifications for each model. 30
Table 3.3: Mean test F_{micro} scores (as percentages) with standard deviations across 20 random seeds for each model and dataset. 33
Table 3.4: Mean test F_{macro} scores (as percentages) with standard deviations across 20 random seeds for each model and dataset. 34
Table 3.5: Mean test F_{micro} and F_{macro} scores (as percentages) with standard deviations across 20 random seeds for the models with fixed feature networks. 35

Table 4.1: Class count and greatest-to-smallest ratio of class prevalence for each task (data element to be extracted). 45
Table 4.2: Mean test F_1-micro scores (as percentages) with standard deviation for each model. 49
Table 4.3: Mean test F_1-macro scores (as percentages) with standard deviation for each model. 50

Table 5.1: Test-RMSE (mean and std. dev. across 20 train-test splits) on UCI datasets, using deep ensembles (DE), deep kernel learning (DKL) and deep ensemble kernel learning (DEKL). 64
Table 5.2: Test-NLL (mean and std. dev. across 20 train-test splits) on UCI datasets, using deep ensembles (DE), deep kernel learning (DKL) and deep ensemble kernel learning (DEKL). 65

Table 1: Mean test F_{micro} scores (as percentages) with standard deviations for each model and dataset. 120
Table 2: Mean test F_{macro} scores (as percentages) with standard deviations for each model and dataset. 121

List of Figures

Figure 1.1: VGG-16 network architecture, a composition of many simple convolutional layers. . . . 3
Figure 1.2: Three main paradigms for combining GPs and DL. . . . 5

Figure 2.1: Shallow-wide CNN architecture. . . . 16

Figure 3.1: Shallow-wide CNN architecture. . . . 23
Figure 3.2: t-SNE visualizations of the 20-Newsgroups test set passed through the feature networks of four different models trained on 20-Newsgroups-100. 36
Figure 3.3: Accuracy-rejection curves (ARCs) for the best models (DKL-lin and DKL-LSC) and the CNN-1 baseline for each dataset. . . . 38

Figure 4.1: Validation F_1-micro (solid) and F_1-macro (dashed) scores of the deterministic (red) and Bayesian (blue) CNN (top) and HiSAN (bottom) models without word2vec on the histology (left) and subsite (right) tasks during training. . . . 51

Figure 5.1: Training various neural architectures considered in this work. . . . 57
Figure 5.2: Synthetic cubic dataset. . . . 62
Figure 5.3: Correlation matrices of learners averaged over all train-test splits on the Kin8nm and Naval propulsion datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). . . . 67

Figure 6.1: Depiction of various NNGP architectures with three predictors (green nodes) and two response variables (red nodes). . . . 72
Figure 6.2: Visualization of the simulated Rings dataset. . . . 81
Figure 6.3: Marginal log-likelihoods (MLL) of three datasets normalized by number of observations (data points) under a bottleneck NNGP for various bottleneck widths and post-bottleneck depths. . . . 83
Figure 6.4: Empirical estimates with standard errors of the quadratic correlation of the outputs of an example bottleneck NNGP with one bottleneck surrounded by two infinite hidden layers, compared to theory (Eq. (6.12)) for various widths of the bottleneck layer. . . . 87
Figure 6.5: Empirical quadratic correlation of the outputs of an example bottleneck NNGP with 11 hidden layers, some of which are restricted to regularly spaced bottlenecks. . . . 88

Figure 6.6: Theoretical quadratic correlation between outputs of a single-bottleneck NNGP over the post-bottleneck depth D for various angles α between the inputs. 90
Figure 6.7: The infinite-depth quadratic correlation between outputs as a function of the weight-variance hyperparameter v_w for an example bottleneck NNGP for various angles α between the inputs. 92
Figure 6.8: Characteristic depth scale of the convergence of the quadratic correlation between outputs to its infinite-depth limit as a function of the weight-variance hyperparameter v_w. 94
Figure 6.9: The infinite-depth quadratic correlation of a single output as a function of the v_w hyperparameter for an example bottleneck NNGP. 96

Figure 1: Mean test RMSE (across all train-test splits) of the DE (red, solid), DEKL-I (green, dotted), and DEKL-O (blue, dashed) models on nine UCI datasets as a function of the number of learners. 125
Figure 2: Mean test NLL (across all train-test splits) of the DE (red, solid), DEKL-I (green, dotted), and DEKL-O (blue, dashed) models on nine UCI datasets as a function of the number of learners. 126
Figure 3: Correlation matrices of learners averaged over all train-test splits on the Boston housing, Concrete, and Energy datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). 127
Figure 4: Correlation matrices of learners averaged over all train-test splits on the Kin8nm, Naval propulsion, and Power plant datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). 128
Figure 5: Correlation matrices of learners averaged over all train-test splits on the Protein, Wine, and Yacht datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). 129

Chapter 1

Introduction

Deep learning (DL) has emerged as the leading paradigm for predictive modeling in a variety of domains, especially those involving large volumes of high-dimensional spatio-temporal data such as images and text [84, 9, 97]. At the heart of the DL approach is a parametric model called a "neural network" (NN) that usually consists of an alternating composition of affine transformations and nonlinear "activation functions" and that is trained by optimizing a loss function with stochastic gradient descent (SGD) or some modern variant of it. One of the most powerful features of the DL paradigm is modularity and composability, where a zoo of various NN architectures can be generated by composing simple building blocks or "layers" in different combinations. This feature also allows practitioners to incorporate various forms of inductive bias into NNs by restricting the network topology based on intuition about the problem at hand and then quickly testing their hypothesis empirically on real datasets. This flexible paradigm led to the development of various convolutional NN architectures such as the VGG-16 (Fig. 1.1) that have found great success in computer vision and attention mechanisms that are now ubiquitous in natural language processing (NLP).

Given the great success of DL and machine learning (ML) in general, the Department of Energy (DOE) now has considerable interest in scientific machine learning (SciML)– the research, development, and application of ML methods for scientific and engineering problems, such as using ML to find patterns in large volumes of scientific data or developing a data-driven model that is more efficient than one based on first principles [12]. However, scientific applications pose unique challenges for ML; for example, it is common in science to ask questions such as: How reliably will a given algorithm work? How robust is a certain solution that is based on a data sample? How rigorously have assumptions and results been verified and validated? In ML, however, answering these questions can be challenging.

In response to these challenges of SciML, the Office of Scientific and Technical Information (OSTI) published a report in 2019 that outlines six priority research directions (PRDs) on foundations and capabilities needed in SciML [12]. One of the PRDs focuses on applications involving complex high-dimensional noisy data that is either limited or is time-consuming or expensive to collect or generate. The PRD specifically highlights the need for probabilistic (e.g., Bayesian) ML methods that are robust to overfitting, reliably quantify parameter uncertainty, and scale well to high-dimensional data. In this thesis, we focus on this PRD because of (1) its applicability to observed, experimental, and simulated data and (2) possible implications for other PRDs such as interpretability and robust validation of ML results.

One example application of SciML arises as part of the National Cancer Institute's (NCI's) cancer surveillance program, which aims to track and understand trends in cancer incidents and mortality across the United States[1]. Cancer pathology reports are the primary source of pertinent information for this surveillance effort, and thus a central task in this endeavor is to extract key data elements from these reports such as tumor site and histological type. Given the national scale of this activity, developing a model that can automatically extract key data elements from cancer pathology reports is imperative for a cost-effective and real-time surveillance program. In 2016, the NCI partnered with several DOE national laboratories to develop such a model; currently, DL is the state-of-the-art methodology for tackling this information extraction task (see Sec. 2.6). However, given the difficulty of the task and the high parameter complexity of the DL architectures that are required, the risk of overfitting

[1] https://seer.cancer.gov

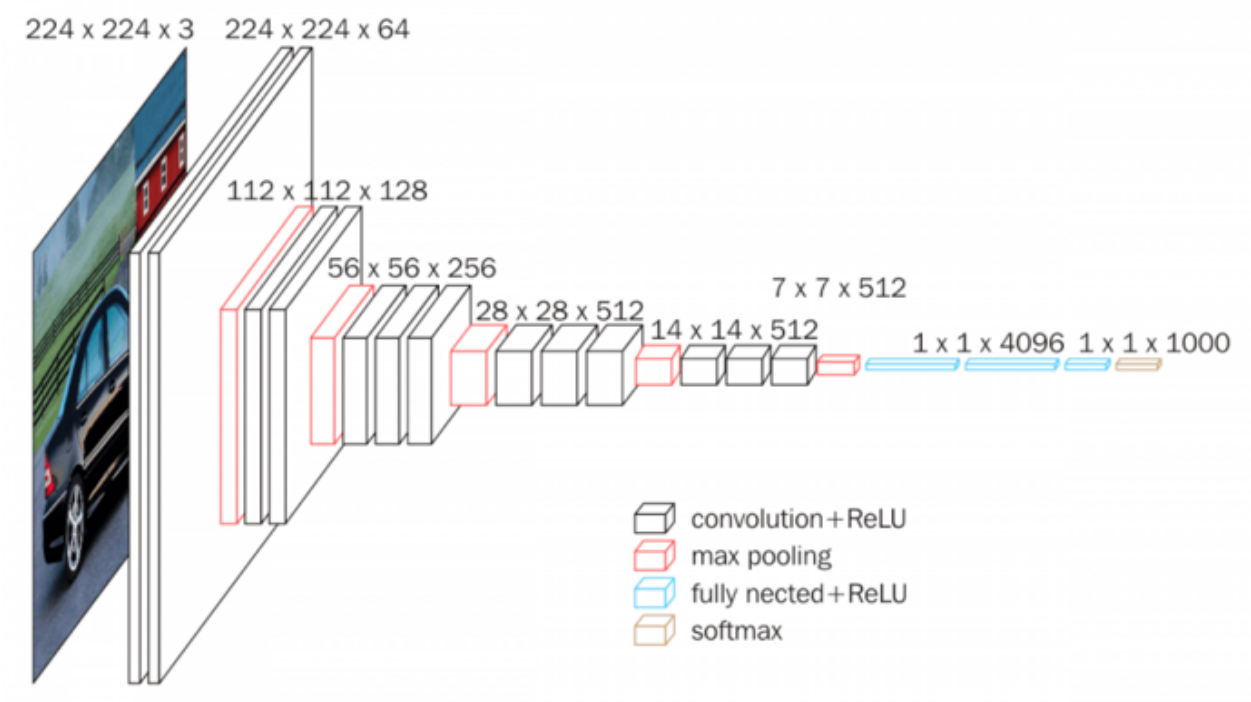


Figure 1.1: VGG-16 network architecture, a composition of many simple convolutional layers. (Source: https://neurohive.io/en/popular-networks/vgg16/).

cannot be ignored and thus there remains a need for a robust model with reliable uncertainty quantification (UQ) capability.

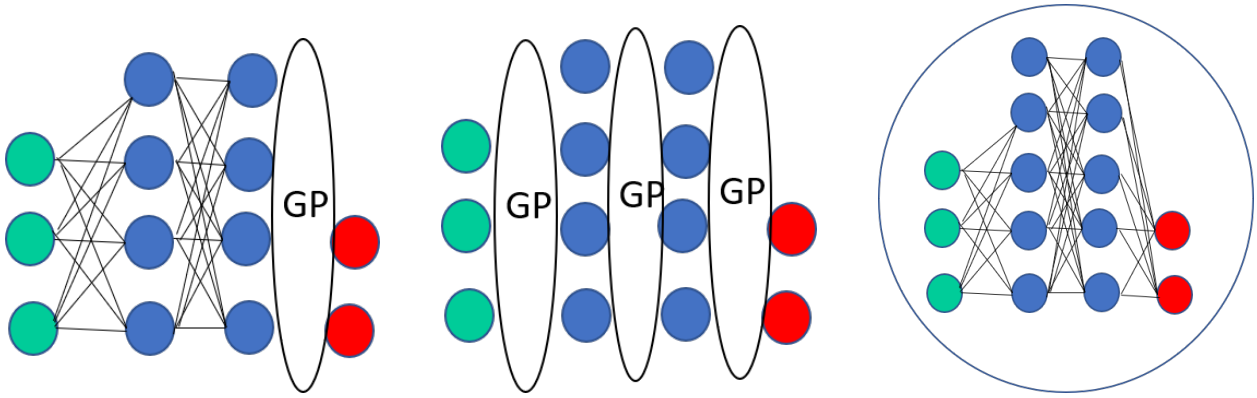
In this thesis, we develop Bayesian DL methods for scientific applications with the aim to improve robustness on high-dimensional noisy data that is limited in size relative to its complexity. Although the Bayesian methodology is well-established for traditional parametric models such as linear regression [55], extending it to DL models is nontrivial; Bayesian neural network (BNN) inference is in general intractable, and approximate sampling-based inference methods scale poorly to large BNN architectures with possibly millions of parameters due to the curse of dimensionality [114]. Moreover, BNNs carry a large number of "weight symmetries", where different parameter configurations correspond to the same input-output function; this makes it challenging to understand how parameter distributions translate into predictive distributions in function space, which is the more natural setting for predictive modeling.

The issues of parameter complexity and symmetries can be circumvented by moving to nonparametric Bayesian methods, where inference is performed directly on functions– or rather, on function values conditional on inputs. The nonparametric approach may be appropriate when the parametric alternative requires more parameters than data points, which is often the case in DL.

One of the most popular nonparametric Bayesian methods in ML is the Gaussian process (GP) [125], which informally speaking, places a Gaussian prior on function space. Unlike BNNs, GP inference admits a closed-form posterior that is itself a GP. A GP prior is usually specified through its "kernel", which determines the function space on which the GP is supported as well as which functions are likelier than others apriori. Analogous to network architecture in DL, the main work of GP methods is in building a kernel with the right balance of inductive bias and expressivity.

A recent strategy for incorporating powerful inductive bias into GP kernels is to combine DL—which has proven itself to be a powerful paradigm for specifying inductive bias—with GPs; we will call such hybrid models GP-DL models. Perhaps the three main GP-DL paradigms in the literature are deep kernel learning (DKL), deep Gaussian processes (DGPs), and neural network Gaussian processes (NNGPs), illustrated in Fig. 1.2 (see Secs. 2.3-2.5 for more background). Each paradigm has its own costs and benefits, resulting from the trade-off between (1) striking a balance between flexibility and robustness and (2) efficiency.

In this thesis, we ask the following question: How can we combine the GP and DL methods, leveraging the strengths of both, and are the resulting GP-DL hybrid models useful in practice? We investigate the three GP-DL paradigms described above, looking at applications to real-world datasets as well as deriving new methods based on these paradigms. In Chapters 3-4, we develop both DKL as well as BNNs for text classification and in particular apply them to the problem of information extraction from cancer pathology reports– the main scientific application considered in this thesis. We note that although BNNs are parametric models, they are a generalization of certain simple DKL models and a special case of DGPs. In Chapters 5-6, we introduce two new methods that extend the existing GP-DL paradigms– deep ensemble kernel learning (DEKL) and the bottleneck NNGP. DEKL is a more efficient and often more powerful extension of DKL, while the bottleneck NNGP is a unification of the DGP and NNGP paradigms that circumvents the parameter complexity of DKL, is



(a) Deep kernel learning (DKL)

(b) Deep Gaussian process (DGP)

(c) Neural network Gaussian process (NNGP)

Figure 1.2: Three main paradigms for combining GPs and DL. In DKL, the final layer of an NN is replaced with a GP (oval). A DGP goes further, with every layer a GP. In contrast to both, the NNGP emerges as a limit of a sequence of BNNs.

better integrated with DL compared to DGPs, and is richer than (no-bottleneck) NNGPs. By developing new GP-DL methods while also evaluating the applicability of existing ones, this thesis advances the budding area at the intersection of nonparametric Bayesian methods and DL with an eye towards robust SciML.

Chapter 2

Background

2.1 Bayesian Neural Networks

The most direct approach to Bayesian DL is simply to place prior distributions on the parameters of an NN and then infer a posterior distribution using Bayes' Rule; such a model is called a "Bayesian neural network" (BNN). Because the BNN makes predictions for all parameter values under the posterior distribution—instead of relying on a point-estimate of the parameters—the BNN exhibits the robustness that we desire from Bayesian DL. Unfortunately, exact BNN inference is intractable and must be approximated.

Perhaps the most popular framework for approximate BNN inference is currently variational inference (VI), where we seek the member of some variational family of distributions that is closest (in terms of KL divergence) to the true posterior. In practice, this amounts to recasting Bayesian inference as an optimization problem where we maximize the "evidence lower bound" (ELBO) with respect to the variational parameters. The original BNN VI method considered only small networks for which the ELBO admitted a closed-form expression [118, 71]. More recently, however, VI has been scaled to larger BNNs by approximating the ELBO with stochastic integration, meaning that the variational model must be sampled at each iteration of SGD [56, 19]. The variational model is therefore typically chosen to be a simple distribution– most commonly a joint normal distribution with diagonal covariance; VI with this choice for the variational model is called mean field variational inference (MFVI), since we assume that the model parameters are independent under the posterior. However, there is some work showing that correlations among parameters can be efficiently modeled if parameterized correctly and that including these correlations in the variational distribution improves performance [100, 135]. Another related idea is Monte Carlo (MC) dropout proposed by Gal and Ghahramani [48], who showed that the popular dropout method in DL [70, 134] can be interpreted as a mixture-of-deltas variational distribution for a BNN; since dropout masks entire weight vectors, this variational distribution also models some dependence among weights. All of these VI approaches, however, require sampling from parameter space (under the variational distribution) and therefore do not scale well to large network architectures containing millions of parameters.

Some recent works have proposed strategies to help reduce the sampling complexity mentioned above. For example, Kingma et al. [82] proposed the local reparameterization trick for linear layers, where instead of sampling the parameters, we directly sample the Gaussian marginal distributions of the (preactivation) outputs of the layer. This method is less efficient for convolutional layers, where the outputs are correlated. Instead, convolutional layers can be handled with the recently introduced flipout method [150]. Ideally, we would draw independent parameter samples for each input into the BNN, but this is inefficient. On the other hand, drawing a single sample for an entire batch of inputs leads to high variance, which is one limitation of traditional MFVI methods for BNN inference. In flipout, all inputs and outputs of a layer are multiplied by IID draws from $\{-1, 1\}$, which helps to reduce correlation across the batch of samples. In this thesis, we use both the local reparameterization trick and flipout when implementing BNNs.

Despite some methods to reduce sampling complexity, BNNs still suffer from two key problems, both of which relate to limitations of the parametric viewpoint. First, there is reason to believe that the true posterior is highly multimodal, and thus it is unclear if a Gaussian variational distribution is sufficiently powerful to accurately model the posterior [63].

Second, there is the problem of selecting an appropriate prior. The prior (as well as the variational distribution) is typically defined in "weight space"– the space of parameter values. However, in predictive modeling, the parameters are unimportant, and we are only interested in the function that the parameters define. NNs exhibit a large number of symmetries in the sense that many parameter configurations define the same function [63, 27, 86]. as a result, how a Gaussian prior in weight space translates to function space is unintuitive, and thus it is unclear how one should go about selecting a weight prior that has desirable properties in function space. Recently, functional variational inference (FVI) was proposed to circumvent these issues of the parametric viewpoint by placing the prior and variational distributions directly on function space [136]. However, because a Lebesgue measure does not exist on a function space of infinite dimension, the ELBO is ill-defined, and its approximation requires drawing either adversarial or random out-of-distribution samples from input space, which could be high-dimensional. Moreover, the variational distribution remains a mean-field BNN. An alternative is to instead start with a model that is naturally function-centric, such as GPs, which we discuss next.

2.2 Gaussian Processes

A Gaussian process (GP) is a nonparametric Bayesian model that can be understood as a random function f such that given any finite collection of inputs $x_1, \ldots, x_N$, the random function values $f(x_1), \ldots, f(x_N)$ follow a joint normal distribution [125]. A real-valued GP on an input space X is completely determined by a mean function $\mu : X \mapsto \mathbb{R}$ and a covariance function or "kernel" $k : X \times X \mapsto \mathbb{R}$ that are used to specify the mean vector and covariance matrix of the GP when restricted to a finite set of inputs. Since the mean function of a GP prior is often set to zero, the kernel completely characterizes the structure of the prior. Once a GP prior is specified over a space of functions, Bayes' Rule can be applied to infer a GP posterior distribution over the same space of functions. In contrast to BNNs, GPs offer a number of advantages:

1. By being nonparametric, they allow us to circumvent the issues of exploding parameter complexity and weight symmetries.

2. A GP prior is placed directly on functions, saving us from having to figure out how a prior over model parameters translates to one over function values and thus making the specification of priors more intuitive.

3. GP inference with a Gaussian likelihood admits a closed-form expression for the posterior, which is itself a GP.

As Bayesian models, GPs maintain the advantages of robustness when data is limited and UQ. GP methods have therefore traditionally been applied to small datasets with only a few thousand examples. However, there are great efforts to extend GP methods to the big data setting, where their advantages may still prove beneficial if overfitting remains an issue– e.g., if a classification problem is sufficiently difficult. There are three main challenges to the application of GPs to big data, all being active areas of research:

1. Applying GPs to large volumes of data: Exact GP inference involves a Cholesky decomposition that has complexity $O(N^3)$ in the number of data points N, and it is therefore nontrivial to perform GP inference for more than a few thousand data points.

2. Applying GPs to high-dimensional data: The most common kernel used in GPs is the radial basis function (RBF) kernel, which is an example of a "local kernel". Local kernels have been shown to suffer from the curse of dimensionality; they degenerate to one-nearest-neighbors regressors and classifiers in very high dimensions and therefore generalize poorly in such settings [14].

3. Incorporating more sophisticated inductive bias into GPs: The specification of a GP prior boils down to the choice of a kernel, as it completely determines the expressivity of the GP as well as how it will generalize to unseen data. It is therefore crucial to select a kernel that carries good generalization properties for the problem at hand.

These are the three challenges that must be overcome in order for GP methods to become comparable to DL.

There is a large body of literature on scaling GP inference to large-volume datasets [99]. The state-of-the-art scalable GP method is the "sparse variational Gaussian process" (SVGP), which approximates the GP posterior in the framework of VI [139, 64, 66]. More specifically, if we want to perform SVGP inference given a training set (X, Y) of N points, then we learn a smaller dataset (Z, U) of M points ($M \ll N$) such that a GP fitted to (Z, U) best explains (X, Y); here the elements of Z are called inducing points and U inducing variables. We do so by approximating the posterior $P(U \mid Y)$ with a Gaussian variational distribution $Q(U)$– i.e., minimizing their KL divergence and deriving an ELBO. The ELBO can then be optimized with SGD or some related gradient-based method. The SVGP approach can be made to scale to even billions of data points if the inducing points are taken to lie on a regular grid and the kernel is factorizable over input dimensions, leading to symmetries in the covariance matrix that can be exploited for fast matrix computations [152, 73]. Finally, GP inference—including approximate SVGP inference with SGD—admits parallelization, and thus specialized GP inference libraries have been built that can take advantage of GPU acceleration– e.g., GPflow, which is based on Tensorflow, and Gpytorch, which is based on Pytorch [39, 53]. GP inference has also been implemented in distributed computing environments [50, 144].

Although the above approaches such as SVGP address scalability with respect to sample size, they do not address the issue of high-dimensional inputs. Since the RBF kernel and its variants work best in low dimensions, the primary strategy to addressing this challenge is based on the assumption that the data manifold has low intrinsic dimension. Given this assumption, the idea is to first map all inputs into a low-dimensional latent feature space and then apply a GP on this space; this approach is generally known as GP manifold regression in the literature [162, 25, 57]. A more recent extension of this idea is deep kernel learning (DKL), where the nonlinear function mapping inputs into the low-dimensional latent space is represented by an NN [154]; we discuss DKL in more detail in Sec. 2.3.

Finally, at the heart of GP methods is the specification of the kernel, which plays a role analogous to the feature extraction network of a DL model. The goal is to design kernels that are expressive (i.e., the GP sample space contains flexible functions) while simultaneously limiting the expressivity through inductive bias that informs how the GP should generalize

to unseen data. Given the success of the DL paradigm to encode inductive bias in network architectures that still remain highly expressive through high parameter complexity, a major approach to designing new kernels is to take inspiration from DL. We mention three such approaches to combining GP and DL methods—deep kernel learning, deep Gaussian processes, and neural network Gaussian processes—which we proceed to discuss next.

2.3 Deep Kernel Learning

Deep kernel learning (DKL) is perhaps the most direct approach to incorporating DL inductive bias into a GP model [154]. Given a GP with kernel $k : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ (such as the RBF) and an NN $\phi : \mathbb{R}^M \mapsto \mathbb{R}^D$, we can construct a "deep kernel" by the composition

$$k_{\text{deep}}(x, x') = k(\phi(x), \phi(x')).$$

The NN parameters are then interpreted as additional kernel hyperparameters that are optimized through maximum marginal likelihood or some approximation to it. Observe that the input dimension M of the deep kernel can be large since inputs first pass through the NN before being passed to the base kernel k.

We can take two perspectives on DKL; a DKL model is

1. a GP with a highly flexible kernel, and
2. an NN where the last layer is a GP.

The first perspective clarifies that a DKL model is a bona fide GP model and is therefore amenable to approximate inference methods such as SVGP. In contrast, the second perspective reveals that "most" of the model is a traditional NN for which we still learn only a point estimate; we therefore expect that the benefit of DKL may be limited, at least compared to more encompassing Bayesian methods.

There are two main approaches to implementing and training DKL models: Stochastic variational inference (SVI) [155, 76, 21, 73] and semistochastic block gradient descent [3, 32]. We focus on the former, as it scales better to large datasets. The motivation of this approach is the observation that a DKL model is just a GP with a very flexible kernel and can thus be implemented as an SVGP as discussed in Sec. 2.2. However, it may be inefficient to take the inducing points Z to lie in the input space, which may have high dimension; instead, we take the inducing points to lie in the low-dimensional feature space to which the NN maps and on which the base GP is applied, thus significantly reducing the dimensionality of the inducing points. Note that this strategy becomes necessary when the input space is not Euclidean–e.g., text documents. The rest of the DKL implementation follows that of the SVGP; we obtain an ELBO that we maximize with SGD.

By adapting the NN feature extractor of a DKL model to different inductive biases, we can apply DKL to various domains. For example, DKL has been successfully applied to image classification using a convolutional architecture [155] and to autoregression on time series for autonomous vehicles based on a recurrent architecture [3]. DKL has also been applied to reinforcement learning [157] as well as for quality assessment of machine translation [32]. We note that the latter used only a fully-connected architecture, and to our knowledge, DKL has

not yet been applied to any NLP task actually requiring a nontrivial NN architecture such as convolution or an attention mechanism. Finally, we note that in almost all applications, the NN feature extractor of the DKL model maps inputs to a space of only 10's of dimensions, and the base kernel is almost always the RBF kernel or some variant of it; non-local base kernels that avoid the curse of dimensionality have not been tested with DKL.

2.4 Deep Gaussian Processes

Inspired by the modularity and composability of the DL paradigm, the deep Gaussian process (DGP) was introduced simply as a composition of GPs. Because a GP can be viewed as a wide BNN through random feature expansion, we will refer to the finite-dimensional spaces between the GP layers of a DGP as "bottlenecks". DGPs have indeed been shown to outperform shallow GPs on a variety of regression and classification tasks, but they were originally limited to small datasets with only a few hundred entries [36]. Much of the DGP literature has therefore been dedicated to developing more efficient and scalable implementations [65, 35, 23, 146, 129, 61]. Salimbeni and Deisenroth [129] in particular show that DGPs can be put into a stochastic variational framework as in Hensman et al. [64], allowing the models to be applied to much larger datasets.

DGPs offer the additional advantage that they can model non-Gaussian distributions. For example, they can capture correlation between multiple outputs [146]. In contrast, the outputs of a shallow multi-output GP are by default independent, which can limit predictive performance for multi-output problems. Although methods have been proposed to model correlation in a shallow multi-output GP, such as through linear mixing of latent outputs [20], DGPs capture correlation naturally through shared feature representations in the latent "bottleneck space"—similar to the approach taken with multi-task deep neural networks [127]. DGPs have therefore been applied to problems that can benefit from modeling correlation between multiple outputs, such as multi-task regression [4] and tasks involving partially observed multivariate outputs—i.e., missing values [146].

The DGP prior has been studied by Lu et al. [101], who show that for a single-bottleneck DGP with a single response variable, the prior has heavy tails, in contrast to shallow GPs. However, they only consider a bottleneck of width one and primarily focus on stationary kernels that do not arise from NNGP limits (see Sec. 2.5) with common activation functions.

There is considerable interest in understanding the "deep limit" of DGPs—i.e. when arbitrarily many GPs are composed together. Duvenaud et al. [45] and Dunlop et al. [43] show that DGPs with a certain class of kernels have trivial, pathological, or convergent deep limits, meaning that increasing the depth of a DGP beyond some point is either detrimental to performance or diminishingly beneficial. However, they do not consider NNGP kernels and thus do not analyze deep limits of architectures with both bottlenecks and infinitely many infinite-width hidden layers (see Secs. 2.5).

Although DGPs were inspired by DL, there is little literature concretely establishing their connection. Duvenaud et al. [45] discuss the connection between DGPs and NNs at a high level to motivate studying the deep limit of DGPs with RBF kernels, but the implication for NNs is not treated formally. Gal and Ghahramani [48] consider a DGP where the kernel of each GP layer is an integral as in Williams [151]. They show that a Monte Carlo estimation

of the kernels leads to a BNN approximation of the DGP, where the width of a hidden layer corresponds to the size of the Monte Carlo sample. However, they do not formally verify convergence of a BNN to a DGP in the limit of infinite width.

Not only has the relationship between DGPs and DL not been studied formally, but there is little work on empirical comparisons between the two paradigms. Although highly scalable to large-volume datasets, DGPs are primarily applied to low-dimensional datasets and are mainly compared to other GP methods and not to DL. Recently, however, there has been greater effort in incorporating DL inductive bias into DGPs. For example, convolution can be incorporated into a GP by taking a weighted average of input-translated copies of the kernel [142, 85, 17, 141], and a recurrent DGP can be constructed by essentially substituting the affine transformations in a recurrent NN with GPs [107, 105, 47, 106]. These DGP models outperform GP baselines that do not incorporate these architectures, and the recurrent DGP even outperforms a DKL model with a recurrent NN as its feature extraction network when data is limited. However, these methods still do not compete with traditional convolutional and recurrent DL models. We believe the limitation is because these DGP models incorporate DL only in spirit and do not correspond to traditional DL in any concrete way as does DKL; for example, the convolutional DGP models use (possibly weighted) global average pooling in place of max-pooling and RBF kernels in place of activation layers. Perhaps a DGP that is more similar to DL models may also be more competitive with DL.

2.5 Neural Network Gaussian Processes

Perhaps the most interesting approach to incorporating DL into GP methods is to consider infinitely wide BNNs. A foundational result in the study of BNNs came when Neal [113] showed that a BNN with one hidden layer converges to a GP in the "wide limit"—i.e., as the number of hidden neurons is sent to infinity. Shortly after, Williams [151] derived analytic expressions for the kernel of the GPs corresponding to neural networks with sigmoidal and Gaussian hidden units. These works connected neural networks to the world of Bayesian nonparametrics and kernel methods and thus offered a new perspective to interrogate and probe the behavior of neural networks.

Since the works of Neal [113], Williams [151], new insights into BNNs have steadily emerged. Cho and Saul [28] interpreted a BNN as a feature embedding map and derived the equations for the propagation of a kernel through the layers of a deep NN with rectified polynomial unit activations. Subsequent works built upon these findings to elucidate key theoretical aspects of NNs including expressivity [121], generalization power [62], initialization [37], and trainability [130]. More recently, the original result by Neal [113] has been extended to deep architectures by showing that a deep BNN converges to a GP as the widths of all hidden layers are simultaneously sent to infinity [89, 40]. We refer to GPs that arise from such a limit as "neural network Gaussian processes" (NNGPs). As the work of de Matthews et al. [40] illustrates, this extension is nontrivial; the proof by Neal [113] relies on the Central Limit Theorem, but the assumption of independent and identically distributed (IID) random variables necessary for the Central Limit Theorem does not hold for deep architectures.

Since the extension of the NNGP limit to deep architectures, there have been a number of works establishing and analyzing analogous wide-limit results for more modern NN

architectures that are used in practice today. These include convolutional NNs [54, 117] and in fact any NN architecture that can be represented as a "tensor program", which includes recurrent NNs and attention mechanisms [159, 160]. We note, however, that there is currently no known closed-form expression for the wide limit of max-pooling layers. Alongside these works, new insights into the trainability and generalization power of NNs have continued to emerge, based on the tractable learning dynamics of NNs in the wide limit [75, 92].

One application of the NNGP limit that is of particular note is that it can make the analysis of the "deep limit"—i.e., as the number of hidden layers is sent to infinity—tractable [121, 130, 161, 89]. Poole et al. [121] and Schoenholz et al. [130] show that the correlation between two inputs transformed through an NNGP with bounded activation function has a fixed point at 100% in the deep limit that transitions from stable to unstable (i.e., ordered to chaotic) when the variance of the Gaussian weights are increased beyond a critical value; the network is shown to be highly expressive in the chaotic phase and optimally trainable near the phase transition. In contrast, for networks with rectified linear unit (ReLU) activations, the correlation between transformed inputs has a stable fixed point at 100% regardless of the weight variance, implying that an NNGP with ReLU activation has no discriminative power at infinite depth [89].

The works described above all consider BNNs in the wide limit, and thus the results and insights therein do not apply to neural network architectures that require one or more finite width or "bottleneck" layers. One of the most important classes of neural networks that require bottleneck layers is that of autoencoders [69, 80]. Another example is neural networks with a word embedding layer, which is currently key to the successful application of neural networks to natural language understanding [111]. Both autoencoders and word embedding layers aim to find dense feature representations and therefore depend on low-dimensional spaces. Even for neural network architectures that are not directly meant for dense representation learning, it has still been argued and demonstrated that bottleneck layers perform data compression and therefore help to boost generalization power [138]. Although there is some evidence that NNGPs can outperform simpler GP baselines—e.g., convolutional NNGP vs. RBF GP on image classification—there is little evidence that NNGPs are competitive with traditional DL; although we obtain a tractable GP model in the wide limit, this is also an extreme deformation of the original architecture, which may negatively impact the original inductive bias of the model. Incorporation of more "bottlenecks"—i.e., combining NNGPs with the DGP paradigm—may therefore benefit the performance of such models.

2.6 Information Extraction from Cancer Pathology Reports

The main scientific application we consider in this thesis is information extraction from cancer pathology reports. It is projected that in 2020, there will be approximately 1.8 million new cases of cancer diagnosed and that 606,520 cancer deaths will occur in the United States alone[1]. When a physician suspects that a patient has cancer, a tissue sample is taken and sent to a pathology lab to be examined. A pathologist then performs a detailed analysis of the

[1] https://www.cancer.org/research/cancer-facts-statistics

sample and records all findings in a cancer pathology report [31]. The report usually contains a wide range of information about the tissue sample—such as a gross description of the tissue as seen by the naked eye, a description of the cells seen under a microscope, a diagnosis of the tumor and grade (how abnormal the cells look and how quickly the tumor is likely to grow and spread)— as well as additional information not directly related to the analysis, including the name and birthdate of the patient and information about the pathologist and laboratory. Since they contain comprehensive information about a potential cancer tumor, these pathology reports are essential in cancer diagnosis and the development of a treatment plan.

Cancer pathology reports thus play a central role in the NCI's SEER program for surveilling national cancer trends and statistics. Specifically, SEER cancer registries across the United States collect pertinent information from cancer pathology reports such as patient demographics, primary tumor site, tumor morphology and stage, first course of treatment, and follow-up status[2]. Currently, trained human experts at these SEER cancer registries are tasked with extracting this information manually– a labor-intensive process that requires the human expert to read each pathology report and annotate the key data elements that need to be tracked. Given the high prevalence of cancer in the United States and the goal to perform this surveillance on a national scale, there is an urgent need to develop accurate and effective automated methods that can assist or replace humans in classifying and extracting information from these reports.

Most existing methods for automated information extraction and classification of cancer pathology reports rely on rule-based approaches [26, 115, 149, 91, 156, 116, 30] or traditional machine learning approaches based on n-gram features, such as Naive Bayes, logistic regression, Adaboost, support vector machines, and conditional random fields [104, 96, 158, 164, 166, 147]. However, hand-engineered rules can fail when a document does not adhere to the pre-determined rules, and n-gram features are unable to account for any potentially important linguistic patterns that span beyond a short window of text [120, 165]. More recently, DL has achieved state-of-the-art classification performance on cancer pathology reports by utilizing a flexible network architecture with appropriate inductive bias that automatically extracts salient features from the documents [122, 52, 51, 5]. In particular, the multi-task shallow-wide convolutional NN (CNN) [5] outperforms a number of traditional machine learning models on the tasks of extracting five key data elements (tumor site, histology, laterality, behavior, and grade) on a dataset of nearly $100,000$ cancer pathology reports. In this CNN architecture (visualized in Fig. 2.1), the words in the input document are first embedded into a Euclidean space where distance reflects semantic similarity; the word-embedded document is then passed through three convolutional filter banks in parallel with respective widths of 3, 4, and 5, which scan the document for certain salient features regardless of location in the document; after a nonlinear activation, a global max-pooling layer is applied to impose translation invariance on the detected features; finally, these extracted features are then passed into a linear multi-task classifier [79].

In this thesis, we place the CNN model in a Bayesian framework in order to boost its performance further by reducing the risk of overfitting. We also do the same for more modern architectures such as those based on attention mechanisms [51].

[2] https://seer.cancer.gov/about/overview.html

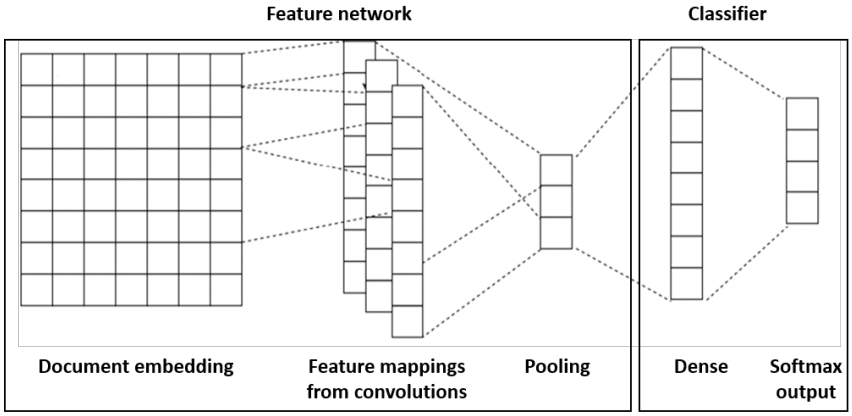


Figure 2.1: Shallow-wide CNN architecture (modified from Figure 1 in [122]). The essential component is the feature network. The linear (i.e., dense) softmax classifier can be replaced by a deeper neural network or GP (see Secs. 2.3).

Chapter 3

Deep Kernel Learning for Medical and Non-Medical Text Classification Tasks

Disclosure

A version of this chapter exists as a manuscript to be submitted to a journal in the future. The authors of the manuscript are Devanshu Agrawal, Abhishek Dubey, Georgia Tourassi, and Jacob Hinkle.

Devanshu Agrawal and Jacob Hinkle conceptualized the paper. Devanshu Agrawal, Abhishek Dubey, and Jacob Hinkle developed the experiments. Devanshu Agrawal implemented and ran the experiments. Devanshu Agrawal wrote the manuscript with support from Abhishek Dubey, Georgia Tourassi, and Jacob Hinkle. All authors read and approved the manuscript.

Abstract

Pathology reports comprise a rich source of data for cancer surveillance across the United States. Cancer registries manually extract key pieces of information from these reports including tumor site, histology, laterality, behavior, grade, and metastatic status. Automating this task is critical for a cost-effective and real-time cancer surveillance program. Convolutional neural networks (CNNs) have demonstrated impressive performance on this information extraction task. However, CNNs are prone to overfitting when training examples are scarce and have limited uncertainty quantification (UQ) capabilities– e.g., abstaining from decision-making when not sufficiently confident. Deep kernel learning (DKL) is a recently introduced probabilistic paradigm that hybridizes CNN and Gaussian process (GP) models, which tend to be more robust to overfitting and offer a more principled framework for UQ. A DKL model is obtained by feeding an NN feature extractor into a GP classifier and is trained using stochastic variational inference. In this project, we build a DKL model for extracting primary tumor site information from a dataset of de-identified cancer pathology reports. We also apply DKL to text classification in general on two public datasets. Our study shows that the DKL model often substantially outperforms the state-of-the-art CNN baseline, as it learns better feature representations through Bayesian training that work well even when transferred to a non-Bayesian architecture. We also show that DKL preserves the quality of UQ as it boosts performance on subsets of increasingly more confident predictions, and that it can help to attain target accuracies while abstaining on significantly fewer decisions.

3.1 Introduction

National cancer surveillance is a critical activity for tracking and understanding trends in cancer incidents and mortality across the United States. Cancer pathology reports hold a rich source of pertinent information for this surveillance effort. A key technical task is, therefore, to extract relevant information from these reports including tumor site, histology, laterality, behavior, grade, and metastatic status. Given the scale of the endeavor, the automation of the information extraction task is vital for timely completion. However, automating information extraction from unstructured text such as pathology reports is challenging because text corresponding to the same concept can be highly variable, such as the description of a tumor's specific location on an organ.

Deep learning (DL) has recently garnered significant attention for its state-of-the-art performance on a variety of natural language processing (NLP) tasks [79, 163]. By casting the information extraction task as a text classification problem, DL has been successfully applied to cancer pathology reports, significantly outperforming other traditional machine learning approaches [122, 52, 51, 6]. In the DL paradigm, a classification model is understood as a composition of a multistage feature extraction pipeline and a classifier that is end-to-end differentiable and hence trainable with gradient-based optimization methods. The key to the success of DL is its ability to extract rich, generalizable feature representations from input documents.

Although DL has achieved state-of-the-art performance on the cancer pathology reports classification problem, several challenges remain. DL models are known to require large amounts of training data for good performance and are otherwise prone to overfitting. Moreover, even as pathology reports datasets grow in volume – presumably leading to more stable performance of DL models – there is often a significant class imbalance in data, so that the reports belonging to underrepresented classes may still fall in the low-sample regime; the need for models more robust than traditional DL in the low-sample regime remains relevant. Finally, uncertainty quantification (UQ) is a critical feature of any clinical information extraction method, although UQ remains an open problem as DL classification models are often overconfident in their predictions [58].

Gaussian processes (GPs) are Bayesian predictive models that perform well on small datasets and also offer a natural framework for UQ as they return full predictive posterior distributions [125]. A GP specifies a distribution over a class of "latent functions" through a covariance function or "kernel", which describes the covariance between function values at any two inputs. Bayes' Theorem is then used to infer the predictive distribution of outputs at new input locations given a set of training inputs. Unlike non-Bayesian classifiers such as neural networks, which make predictions based only on the most likely set of model parameters given data, a GP model returns a distribution of predictions from all latent functions. If the most likely function (i.e., posterior mode) overfits to the training data and misclassifies a test input, other less likely functions may still classify it correctly. The predictive posterior mean, which takes into account the predictions of all latent functions of the GP, is therefore often more robust to overfitting than the mode alone. The kernel determines the expressiveness of the GP model through a prior distribution on these latent functions and therefore plays a similar role to the feature extractor part of a DL model. This connection has led many researchers to wonder if DL can be used to derive new, richer kernels for more powerful GP models.

There is a large body of work that connects and combines GP and DL models in a meaningful way, with many works showing that the connection is even fundamental [113, 151, 89]. One of the simplest approaches is "deep kernel learning" (DKL) [154, 155, 76, 21, 73], where inputs are first mapped to a latent feature space through a DL network, and a GP model is then built on the latent space. The DL network can be fully encapsulated in the GP kernel with the DL weights interpreted as kernel hyperparameters– hence "deep kernel". The entire DKL model can then be trained by maximizing the marginal likelihood as is typically done for GP models. DKL has been shown to outperform traditional DL and GP methods on problems such as image classification [155] and has also been successfully applied to autoregression on time series [3] and reinforcement learning [157]. Besides an application

to the quality estimation of sentence-level predictions [32], we are unaware of any other applications of DKL to the NLP domain.

In addition to boosting predictive performance, DKL may also be able to improve the UQ capabilities of a model. By UQ, we specifically mean the ability of a classification model to abstain from decision-making when it is not sufficiently confident in its prediction, in which case the decision may be defered to a human expert. A model can therefore remain useful as long as it learns to assign high confidence only to predictions that are likely to be correct and abstains on only a small number of low-confidence predictions. The key limitation of traditional DL classification models for UQ is that they measure confidence in terms of distance from the decision boundary, and may therefore assign high confidence to outliers far from observed data. On the other hand, GP classifiers – including DKL – are capable of outlier detection and will appropriately assign low confidence to such extreme examples, assuming the right kernel function is used (or learned, in the case of DKL). Although other UQ methods such as Bayesian neural networks [114, 77] and Monte-Carlo dropout [48] are also formulated in a Bayesian framework, they require sampling from parameter space, whose dimension can easily be in the millions. In contrast, since DKL is only partially Bayesian, it could offer the right balance of precise and scalable UQ.

In this project, we develop a DKL model for extracting primary tumor site information from cancer pathology reports. We also test this DKL model on text classification tasks in general, specifically on the publicly available IMDB Movie Reviews and 20-Newsgroups datasets. For the feature extraction network of the DKL model, we use a shallow-wide convolutional neural network (CNN) architecture that was specifically developed for text classification [79] and has already been successfully applied to cancer pathology reports [122]. The DKL model often outperforms the baseline CNN, and we show that this is because the DKL model is able to extract better feature representations that work well even with non-Bayesian classifiers. Moreover, the DKL model often maintains superior performance as more low-confidence predictions are rejected, thus preserving UQ capabilities. Our main conclusion is that DKL is relevant for text classification as it offers an effective and elegantly simple framework to hybridize DL and Bayesian modeling.

3.2 Datasets

3.2.1 Electronic Pathology Reports

We used a dataset of 942 de-identified electronic pathology reports (EPR) gathered from five different SEER cancer registries (CT, HI, KY, NM, Seattle). The reports were manually labeled by the cancer registries using standard ICD-O-3 topography codes for encoding primary tumor site as per the SEER coding guidelines. The topography codes for this EPR dataset correspond to 7 breast and 5 lung topography sites. Descriptions of these 12 topography codes as well as their observation counts can be found in Table 1 of our previous work [122]. We note that there is heavy class imbalance, with 6 of the codes having counts fewer than 50. For this study, we only used the top six most prevalent classes, which are more balanced. We will call this dataset EPR.

The reports were preprocessed from their original XML format and tokenized to integer word indices in identical fashion as in Qiu et al. [122], Dubey et al. [42]. Each tokenized report was either zero-padded or truncated to a length of 1500– the 99th percentile of the report length distribution.

3.2.2 Public Text Datasets

We considered two additional publicly available datasets that are common benchmarks for text classification– IMDB Movie Reviews [102] and 20-Newsgroups [41]. The IMDB dataset consists of 50000 movie reviews, each of which has been labeled based on apparent sentiment as either positive or negative. The dataset has 25000 training and 25000 test documents and has balanced classes. The 20-Newsgroups dataset contains 18846 newsgroups posts on 20 different topics. This dataset has 11314 training and 7532 test documents and is nearly class-balanced with relative class-label frequencies between 0.033 and 0.053. The classes are stratified between the training and test sets so that relative class-label frequencies are the same in the test set as in the training set.

We used the pretokenized IMDB dataset shipped with the Keras library, and we tokenized the 20-Newsgroups dataset using the default regular expressions set in the Keras tokenizer. We restricted the vocabularies of IMDB and 20-Newsgroups to the top 5000 and 10000 most frequent words respectively, which account for about 90% of all words in their training sets. We either zero-padded or truncated the documents in IMDB and 20-Newsgroups to have lengths of 1000 and 2000 respectively, which are the 99th percentiles of the respective document length distributions.

We were interested to see the effect of training sample size on DKL and therefore tested our models on various subsampled versions of IMDB and 20-Newsgroups. We used 22% and 100% of the 20-Newsgroups training set (20Newsgroups-22 and 20Newsgroups-100 respectively) and 1%, 5%, and 100% of the IMDB training set (IMDB-1, IMDB-5, and IMDB-100 respectively). We chose the percentages such that larger datasets are subsampled to have an average number of training points per class that matches that of smaller datasets (Table 3.1). We sampled without replacement and with stratification to maintain relative class-label frequencies. We did not subsample any of the test sets.

3.3 Classification methods

3.3.1 Convolutional neural networks

We used as our baseline model a CNN with a shallow-wide architecture first developed for text classification by Kim [79] and later successfully applied to the EPR dataset by Qiu et al. [122]. The CNN architecture is depicted in Fig. 3.1.

The CNN expects a text document $\mathbf{x}$ as its input, which is represented as a sequence of integer tokens x_i, with each token indexing a word in the vocabulary. The input sequence is first passed through a word embedding layer that maps each token to a word vector of dimension E, which is typically much smaller than the vocabulary size V. The embedding

Table 3.1: Number of classes, average number of training points per class, and average number of test points per class for each dataset. The six datasets fall into three sample regimes based on training points per class: 20Newsgroups-22 and IMDB-1 have been subsampled from 20-Newsgroups and IMDB such that they fall in the same regime as EPR, IMDB-5 has been subsampled to match 20Newsgroups-100, and IMDB-100 is in its own regime.

Dataset	Classes	Training points per class	Test points per class
EPR	6	123.75	13.75
20News-22	20	124.45	376.6
IMDB-1	2	125.0	12 500.0
20News-100	20	565.7	376.6
IMDB-5	2	625.0	12 500.0
IMDB-100	2	12 500.0	12 500.0

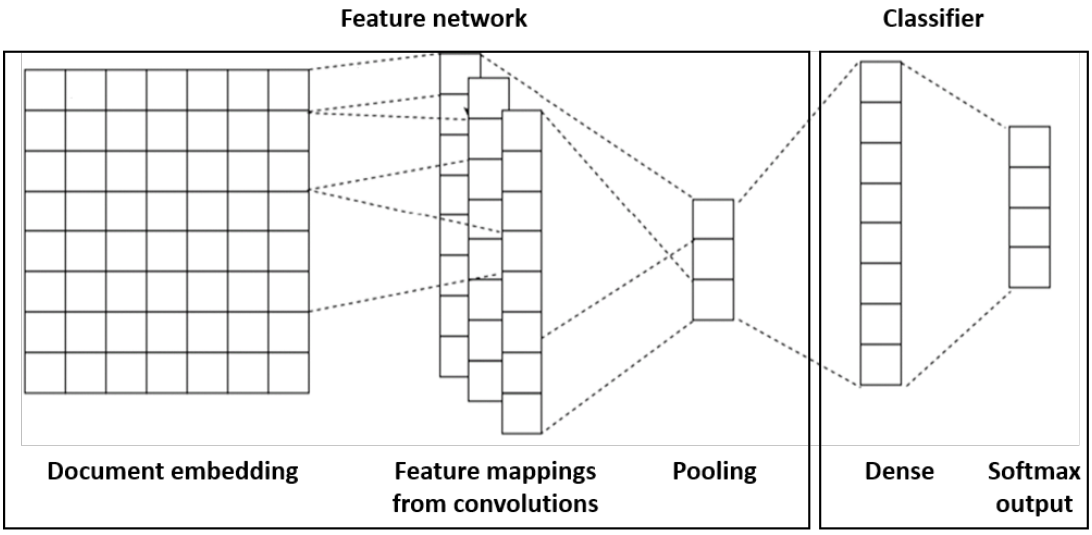


Figure 3.1: Shallow-wide CNN architecture (modified from Figure 1 in Qiu et al. [122]). The essential component is the feature network. The linear (i.e., dense) softmax classifier can be replaced by a deeper neural network or GP (see Secs. 3.3.3 and 3.4.2).

layer sends the document $\mathbf{x}$ to a matrix $\mathbf{H}$ whose elements are given by

$$h_{ij} = \text{embed}(\mathbf{x})|_{i,j} = e_{x_i j}, \tag{3.1}$$

where e_{kj} are the elements of the $V \times E$ word embedding matrix $\mathbf{E}$.

The embedding output is then passed through P convolutional modules of differing "widths" in parallel. A convolutional module with F convolutional filters all of width M returns a vector $\mathbf{z}$ whose elements are defined as

$$z_j = \text{conv}(\mathbf{H}) = \max_i \text{ReLU}\left(\sum_{m,n=1}^{M,E} h_{i+m-1,n} w_{mnj} + b_j\right), \tag{3.2}$$

where w_{mnj} and b_j are elements of the $M \times E \times F$ convolutional weight matrix $\mathbf{W}$ and F-dimensional bias vector $\mathbf{b}$ respectively, and $\text{ReLU}(\cdot)$ is the rectified linear unit (ReLU) activation function. A convolutional module thus extracts salient features localized to M consecutive tokens in the input sequence while respecting translational invariance. The output vectors of all convolutional modules are then concatenated together, producing a final feature representation $\tilde{\mathbf{z}}$ of dimension $P \cdot F$. That is, the architecture up to this point comprises the feature extraction network of the CNN.

The extracted feature representation is then passed through a linear softmax classifier (LSC) defined as

$$\mathbf{y} = \text{softmax}\left(\left\{\sum_{i=1}^{PF} \tilde{z}_i \tilde{w}_{ij} + \tilde{b}_j\right\}_{j=1}^{C}\right), \tag{3.3}$$

where C is the number of classes available as document labels, $\tilde{w}_{ij}$ and $\tilde{b}_j$ are elements of the $PF \times C$ weight matrix $\tilde{\mathbf{W}}$ and C-dimensional bias vector $\tilde{\mathbf{b}}$, and $\text{softmax}(\cdot)$ is defined such that $\mathbf{y} = \text{softmax}(\mathbf{x})$ has elements

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}. \tag{3.4}$$

If we denote the CNN described above as the function $\boldsymbol{\phi}(\cdot; \boldsymbol{\omega})$ with network parameters $\boldsymbol{\omega}$, and if we are given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ of labeled documents with each $\mathbf{y}_i$ a one-hot vector encoding the class label of document $\mathbf{x}_i$, then the CNN is trained by minimizing the cross-entropy loss function defined as

$$\mathcal{L}_{\text{ce}}(\boldsymbol{\omega}) = -\frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i^{\top} \ln[\boldsymbol{\phi}(\mathbf{x}_i; \boldsymbol{\omega})]. \tag{3.5}$$

At test time, the predicted class of an input document is the argmax of the output softmax vector– i.e., the class with the greatest softmax score. We take the softmax score of the predicted class to be the confidence of the prediction.

3.3.2 Gaussian process classifiers

A GP classifier is a classification model obtained by mapping a collection of C independent latent GPs $\mathbf{f} = (f^{(1)}, \ldots, f^{(C)})$ to a probability mass function over C classes through an inverse-link function $\mathbf{g}$:

$$\mathbf{y} = \mathbf{g}(\mathbf{f}(\mathbf{x})). \tag{3.6}$$

A GP with mean function $\mu(\cdot)$ and covariance function or "kernel" $k(\cdot, \cdot)$ is a random function f such that for every finite set of inputs $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$, the corresponding function values $f(\mathbf{X}) = \{f(\mathbf{x}_i)\}_{i=1}^N$ follow a joint normal distribution with mean parameter $\boldsymbol{\mu} = \mu(\mathbf{X})$ and covariance matrix $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$, which have entries $\mu_i = \mu(\mathbf{x}_i)$ and $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ respectively. We denote this by $f \sim \mathcal{GP}(\mu, k)$. Each latent function $f^{(c)}$ in Eq. (3.6) is taken to follow the GP prior $\mathcal{GP}(0, k)$. The choice of kernel k is at the heart of GP methods since it determines the class of functions that are used for modeling. Two of the simplest kernels that are relevant for this work are the radial basis function (RBF) kernel and linear kernel:

$$k_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\lambda^2}} \tag{3.7}$$

$$k_{\text{lin}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \mathbf{x}^\top \mathbf{x}', \tag{3.8}$$

where σ^2 and λ are the variance and lengthscale hyperparameters respectively. The key difference between the RBF and linear kernels is that the former is "local" while the latter is "nonlocal", meaning that the RBF kernel assigns high correlation only to points near to one another while the linear kernel can do so even for distant points. This implies that the RBF and linear kernels can behave very differently especially in high dimensions due to the curse of dimensionality [14]. The inverse-link function $\mathbf{g}$ is designed to "squash" vectors in $\mathbb{R}^C$ to vectors in $[0, 1]^C$, thus yielding a probability distribution over the C classes. The canonical example is the softmax function, although this is not necessarily the best choice for GP classifiers. We discuss inverse-link functions in more detail in Sec. 3.3.2.

Suppose we have a training set $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^D \times \mathbb{R}^C\}_{i=1}^N$ for a classification task where each $\mathbf{y}_i$ is a one-hot vector encoding the class label of input $\mathbf{x}_i$. Given a set of test inputs, $\mathbf{X}_* = \{\mathbf{x}_{*i} \in \mathbb{R}^D\}_{i=1}^{N_*}$, we wish to make test predictions $\mathbf{Y}_* = \{\mathbf{y}_{*i} \in \mathbb{R}^C\}_{i=1}^{N_*}$. Consider the GP classifier model in Eq. (3.6). As shorthand, let $\mathbf{F} = \{\mathbf{f}_i \in \mathbb{R}^C\}_{i=1}^N$ and $F_* = \{\mathbf{f}_{*i} \in \mathbb{R}^C\}_{i=1}^{N_*}$ be the latent GP function values over the training inputs and test inputs respectively, where $\mathbf{f}_i = \mathbf{f}(\mathbf{x}_i)$ and $\mathbf{f}_{*i} = \mathbf{f}(\mathbf{x}_{*i})$. The predictive posterior distribution over the test set is then given by

$$p(\mathbf{Y}_* \mid \mathbf{Y}) = \int p(\mathbf{Y}_* \mid \mathbf{F}_*) p(\mathbf{F}_* \mid \mathbf{Y}) \, \mathrm{d}\mathbf{F}_*, \tag{3.9}$$

where $p(\mathbf{F}_* \mid \mathbf{Y})$ is the GP posterior on the test set, and the likelihood function $p(\mathbf{y} \mid \mathbf{f})$ for any vector $\mathbf{f}$ and one-hot vector $\mathbf{y}$ is given by

$$p(\mathbf{y} \mid \mathbf{f}) = \mathbf{y}^\top \mathbf{g}(\mathbf{f}). \tag{3.10}$$

For classification inverse-link functions $\mathbf{g}$ such as the softmax function, the GP posterior $p(\mathbf{F}_* \mid \mathbf{Y})$ is intractable and must be approximated. Moreover, GP inference involves the

Cholesky decomposition of the covariance matrix $k(\mathbf{X}, \mathbf{X})$, which becomes challenging for more than a few thousand training points.

The sparse variational Gaussian process (SVGP) model addresses both issues by approximating GP inference with the inducing points method in the variational inference framework [139, 64]. In our work, we use the SVGP model, which we briefly describe next. The SVGP introduces a set of additional unknown variables, called "inducing points" to address the scalability issue. It is assumed that all information in $\mathbf{Y}$ relevant to $\mathbf{F}$ is captured through M inducing points $\mathbf{Z} \in \mathbb{R}^D$ and their corresponding latent GP function values $\mathbf{U} \in \mathbb{R}^C$, where $(\mathbf{Z}, \mathbf{U}) = \{(\mathbf{z}_i, \mathbf{u}_i)\}_{i=1}^M$ for some $M < N$. The GP posterior on the training set is then derived with these new variables as

$$p(\mathbf{F} \mid \mathbf{Y}) = \int p(\mathbf{F} \mid \mathbf{U}) p(\mathbf{U} \mid \mathbf{Y}) \, \mathrm{d}\mathbf{U}. \tag{3.11}$$

This shifts the burden of the Cholesky decomposition of $k(\mathbf{X}, \mathbf{X})$ to $k(\mathbf{Z}, \mathbf{Z})$, thus reducing the computation cost from $O(N^3)$ to $O(M^3)$, where the number of inducing points M is typically much smaller than the training set size N. However, the term $p(\mathbf{U} \mid \mathbf{Y})$ remains inexact; understanding it amounts to inferring the augmented posterior $p(\mathbf{F}, \mathbf{U} \mid \mathbf{Y})$, which SVGP approximates through variational inference. The SVGP introduces a variational family of distributions $q(\mathbf{F}, \mathbf{U}; \boldsymbol{\theta})$ and searches for the set of variational parameters and inducing points that minimizes the KL divergence between the variational distribution and target posterior:

$$\hat{\boldsymbol{\theta}}, \hat{\mathbf{Z}} = \mathrm{argmin}_{\boldsymbol{\theta}, \mathbf{Z}} \, \mathrm{D}_{\mathrm{KL}}(q(\mathbf{F}, \mathbf{U}; \boldsymbol{\theta}) \mid p(\mathbf{F}, \mathbf{U} \mid \mathbf{Y})). \tag{3.12}$$

The SVGP model uses the variational family

$$q(\mathbf{F}, \mathbf{U}; \boldsymbol{\theta}) = p(\mathbf{F} \mid \mathbf{U}) \;\; q(\mathbf{U}; \boldsymbol{\theta}), \tag{3.13}$$

$$q(\mathbf{U}; \boldsymbol{\theta}) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{u}^{(c)}; \mathbf{m}^{(c)}, \mathbf{S}^{(c)}), \tag{3.14}$$

where $\mathbf{u}^{(c)} = f^{(c)}(\mathbf{Z})$ and $\boldsymbol{\theta} = \{(\mathbf{m}^{(c)}, \mathbf{S}^{(c)})\}_{c=1}^C$ are the variational parameters (variational means and covariances of the latent function values at the inducing points).

The optimization problem in Eq. (3.12) is equivalent to maximizing the evidence lower bound (ELBO), which in the case of our variational family takes the simplified form

$$\mathrm{ELBO}(\boldsymbol{\theta}, \mathbf{Z}) = \sum_{i=1}^{N} \int \ln[p(\mathbf{y}_i \mid \mathbf{f}_i)] \;\; q(\mathbf{f}_i; \boldsymbol{\theta}) \, \mathrm{d}\mathbf{f}_i \\ - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{U}; \boldsymbol{\theta}) \mid p(\mathbf{U})), \tag{3.15}$$

where $p(\mathbf{U}) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{u}^{(c)}; 0, k(\mathbf{Z}, \mathbf{Z}))$ is the GP prior at the inducing points and $q(\mathbf{f}_i; \boldsymbol{\theta})$ is the marginalization of $q(\mathbf{F}, \mathbf{U}; \boldsymbol{\theta})$ over $\mathbf{U}$ and all $\mathbf{f}_j$, $j \neq i$. We simultaneously maximize the ELBO with respect to the kernel hyperparameters as well.

At test time, the SVGP infers the approximate predictive posterior distribution by the formula

$$p(\mathbf{Y}_* \mid \mathbf{Y}) = \int p(\mathbf{Y}_* \mid \mathbf{F}_*) q(\mathbf{F}_*; \hat{\boldsymbol{\theta}}) \, \mathrm{d}\mathbf{F}_*, \tag{3.16}$$

which gives a multinomial distribution over the C classes, viewed as a C-dimensional vector of probabilities. We take the argmax of this output vector to be the class label prediction and the corresponding class probability to be the associated confidence score.

Inverse-Link Functions

The softmax function in Eq. (3.3) is a natural choice for the inverse-link function in the GP classifier, since it is also commonly used in traditional DL classification models. However, the integral in Eq. (3.15) with a softmax inverse-link function remains intractable; even an efficient approximation remains elusive for a large number of classes C since the number of samples required for the approximation grows exponentially with C. A popular alternative to the softmax function is the "robustmax" function [67], which is defined such that $\mathbf{y} = \text{robustmax}(\mathbf{x})$ has components

$$y_i = \begin{cases} 1 - \varepsilon & \text{if } i = \arg\max(\mathbf{x}) \\ \frac{\varepsilon}{C-1} & \text{otherwise,} \end{cases} \tag{3.17}$$

for some small $\varepsilon > 0$. The robustmax is the default link function for GP classifiers in the popular GP library GPflow [39] and has also been used previously with DKL [21]. The key advantage of the robustmax is that it allows the C-dimensional integral in Eq. (3.15) to be reduced to a one-dimensional integral that can be efficiently approximated using Gauss-Hermite quadrature, thus circumventing the curse of dimensionality entirely. When we do use softmax – e.g., on two-class datasets – we approximate the integrals in Eqs. (3.15)-(3.16) with Monte-Carlo sampling instead of Gauss-Hermite quadrature.

3.3.3 Deep kernel learning

A DKL model is a GP whose kernel encapsulates a neural network for feature extraction [154, 155]. A deep kernel is defined as

$$k_{\text{deep}}(\mathbf{x}, \mathbf{x}') = k(\boldsymbol{\phi}(\mathbf{x}; \boldsymbol{\omega}), \boldsymbol{\phi}(\mathbf{x}'; \boldsymbol{\omega})), \tag{3.18}$$

where k is any kernel (such as the RBF or linear kernel) and $\boldsymbol{\phi}(\cdot; \boldsymbol{\omega})$ is a neural network with weights $\boldsymbol{\omega}$. The kernel hyperparameters of k_{deep} include all hyperparameters of k as well as the network weights $\boldsymbol{\omega}$.

Since a DKL model is just a GP, then it can also be trained using inducing points in the sparse variational framework. However, for high-dimensional inputs such as images or text, having the inducing points lie in the input space may be inefficient or impractical. We instead think of a DKL model as a composition of a neural network $\boldsymbol{\phi}(\cdot; \boldsymbol{\omega})$ that maps inputs into a feature space $\boldsymbol{\mathcal{Z}}$ and an SVGP with kernel k implemented on $\boldsymbol{\mathcal{Z}}$; we take the inducing points to lie directly in the DKL feature space $\boldsymbol{\mathcal{Z}}$, whose dimension is typically much smaller than the input dimension. This DKL model is trained by maximizing the ELBO given in Eq. (3.15), with the only modification being that an input $\mathbf{x}$ in Eq. (3.15) is replaced with $\boldsymbol{\phi}(\mathbf{x}; \boldsymbol{\omega})$ where the network parameters $\boldsymbol{\omega}$ are trainable.

The first DKL model introduced by Wilson et al. [154] made use of an additional approximation by requiring that the inducing points lie on a regular lattice. This regular

structure allowed the covariance matrix to be factored into a Kronecker product of circulant matrices, which facilitates efficient and highly scalable GP inference [152]. However, this approach also means that the number of inducing points grows exponentially with the dimension of the DKL feature space $\boldsymbol{\mathcal{Z}}$ and is therefore restricted to feature dimensions of up to 5-10. In this work, we are interested in a DKL model whose feature network is a shallow-wide CNN as described in Sec. 3.3.1; we replace the linear-softmax classifier with a GP classifier. For a CNN with P convolutional modules each holding F filters, the DKL feature dimension is $P \cdot F$, which is easily on the order of 100. For this reason, we do not impose any regular structure on the inducing points and allow the training data to guide their positions independently; this approach was taken in Bradshaw et al. [21].

3.3.4 Uncertainty quantification

For both the CNN and DKL models, we calculate the confidence of a test prediction as the likelihood of the predicted class. However, the CNN and DKL models encode very different uncertainties in their confidence scores. Since the CNN and DKL models have identical feature network architectures, this difference comes down to the difference between the linear softmax and GP classifiers. For a binary classification task, the linear softmax classifier measures confidence in terms of distance from the decision boundary; test points further from the decision boundary are classified with higher confidence. Even outliers that are distant from both the decision boundary and the training set are classified with high confidence, which is undesirable. A GP classifier, on the other hand, can potentially detect outliers and assign low confidence to their predicted class-labels. The variance of the latent GP value $f^{(c)}(\mathbf{x}_*)$ associated to class c at a test input $\mathbf{x}_*$ under the variational distribution q is given by

$$\mathrm{var}[f^{(c)}(\mathbf{x}_*)] = [k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{Z})\mathbf{a}] + \mathbf{a}^\top \mathbf{S}^{(c)} \mathbf{a} \tag{3.19}$$

$$\mathbf{a} = k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{z}, \mathbf{x}_*), \tag{3.20}$$

where we use the notation introduced in Sec. 3.3.2. With the linear kernel, the first term in Eq. (3.19) measures the squared distance of $\mathbf{x}_*$ from the subspace spanned by the inducing points $\mathbf{Z}$. Test points far from the training set as summarized by the inducing points are therefore assigned high latent variance, which translates to low confidence. DKL could, therefore, offer more reliable UQ than vanilla CNN models.

3.4 Experimental setup

We compare the classification accuracy of DKL models [155, 21] with the state-of-the-art CNN model [79]. We implemented all models in Tensorflow and used the Tensorflow-based library GPflow [39] for the DKL models.[1]

For the EPR dataset, we performed ten-fold stratified cross-validation (CV) and measured test performance on predictions aggregated from all ten test folds. For each of the ten training-test splits, we further split the training fold into training and validation sets, stratified to maintain relative class frequencies. We took the validation set to be 11% of the training set

[1] Code for this project can be found at https://code.ornl.gov/d0a/epath_dkl.

so that it is approximately equal in size to the test fold. For the two 20-Newsgroups and three IMDB datasets (see Table 3.1), we evaluated performance on the full official test sets regardless of what percentage of the training set was used. For each of these datasets, we still used 11% of the training set for validation.

To account for the variability in test performance due to random initialization and train-validation-test splits, we ran each model for 20 different seeds for random number generation (RNG). We believe this is critical to assess whether performance gaps between models are significant or not.

3.4.1 Metrics

We measured the classification performance of all models with the F_1 score– the harmonic mean of precision and recall. We calculated both micro-averaged and macro-averaged F_1 scores denoted by F_{micro} and F_{micro} respectively. The micro-averaged F_1 score measures the overall classification accuracy, while the macro-averaged F_1 score indicates the level of class balance in the predictions as compared to the relative class frequencies in the ground truth labels. For example, the gap between the F_{micro} and F_{macro} scores of a random binary classifier is minimized when the predicted and true class distributions are equal.

We measured the UQ performance of a model in terms of its ability to abstain from decision-making when it is not sufficiently confident in its prediction. For every confidence threshold, we calculate the rejection rate (fraction of predictions whose confidence scores fail to meet the threshold) and accuracy on predictions not rejected, thus sweeping out an accuracy-rejection curve (ARC) [29, 59, 112]. More precisely, we first rank the predictions in order of increasing confidence. We assign correctness scores of 0 and 1 to incorrect and correct predictions respectively. If a set of predictions has the same confidence, then we re-assign their correctness scores to their average, thereby avoiding arbitrary orderings. We then calculate rejection rate and accuracy while looping over the ranked predictions, where accuracy is the average correctness score. A model with good UQ should assign higher confidences to predictions that are more likely to be correct and should therefore exhibit an increasing ARC. If a target accuracy is specified, then the model with the best UQ is the one that achieves the target accuracy with the lowest rejection rate.

3.4.2 Models

We compared six models: CNN-1, CNN-2, DKL-lin, DKL-RBF, CNN-SVGP, and DKL-LSC (see Table 3.2). All models share a common shallow-wide feature network architecture (see Sec. 3.3.1) with embedding dimension 300 and three convolutional filter modules of widths 3, 4, and 5, each containing 128 filters. These models differ only in either their classifiers or training approaches. CNN-1 uses a linear softmax classifier (one dense layer) and is the main CNN baseline. CNN-2 includes a hidden 128-neuron ReLU activation layer before a linear softmax classifier (two dense layers). We include CNN-2 in our experiments to examine whether the performance of the baseline CNN can be further improved by merely increasing the complexity of the classifier. DKL-lin and DKL-RBF use SVGP classifiers with linear and RBF kernels respectively. We used the robustmax inverse-link function on EPR and the two 20-Newsgroups datasets but used softmax on the three IMDB datasets, since IMDB has only

Table 3.2: Classifier, pretraining, and feature network specifications for each model. For the last two models, "Dense layers" and "Kernel" apply to the classifier used at test time, not during training.

Model	Dense layers	Kernel	Options for pretraining	Fixed Features
CNN-1	1		CNN-1, best DKL	
CNN-2	2		CNN-2, best DKL	
DKL-lin		Linear	CNN-1	
DKL-RBF		RBF	CNN-1	
CNN-SVGP		Linear		CNN-1
DKL-LSC	1			best DKL

two classes. We used 256 inducing points on all datasets except for IMDB-1, for which we used 128 inducing points since the training set (excluding validation set) has less than 256 documents.

In these first four models, the feature networks and classifiers are trained together end-to-end. To determine whether the DKL models outperform the CNN models merely because of the SVGP classifier at test time or because the DKL feature network is better than the CNN feature network at feature extraction, we introduced the latter two models. In the DKL-LSC model, we train a DKL-lin model and keep its feature network, and then only train a linear softmax classifier on the extracted features. In the CNN-SVGP model, we train a CNN-1 model and keep its feature network, and then only train an SVGP classifier on the extracted features.

Pretraining is often necessary to train or improve the performance of DKL models [155, 21]. To pretrain a model, we initialize its feature network to the weights of another trained model We briefly undertrain the classifier with the feature network held fixed, but afterward, train the entire network (feature network and classifier) end-to-end. We tested each DKL model both trained from scratch and pretrained with the CNN-1 model (Table 3.2). To examine whether the pretrained DKL models have an "unfair" advantage over the baseline CNN because of longer training duration, we also tested the baseline CNN models both trained from scratch as well as pretrained with copies of themselves; Self-pretraining effectively extends the training durations of the CNN models except that their classifiers are re-initialized half-way through training.

Finally, we tested CNN-1 pretrained with the best DKL model. This model lets us understand how a linear softmax classifier affects (e.g., corrupts or improves) DKL-learned features.

3.4.3 Training details

We initialized the feature network weights of all unpretrained models randomly. We initialized the GP inducing points of the DKL models by passing the training data through the feature network and then performing K-means clustering on the resulting feature vectors; the inducing points are initialized to the cluster means. We trained all models for up to 100 epochs using the Adam optimizer with a minibatch size 32 and learning rate 0.001. We used early stopping with patience 10. For the models with pretrained features, we briefly trained the classifiers for 5 epochs with the feature networks held fixed before training end-to-end for up to 100 epochs. We took the model from the epoch giving the highest validation F_{micro} score for prediction on test sets for evaluation. We trained all models using L_2 regularization on the feature networks with a regularization coefficient of 0.001.

3.5 Results

3.5.1 Model performance

For each model and each of the 20 RNG seeds, we calculated F_1 scores on predictions aggregated from all 10 test folds of the EPR dataset and on predictions made on the official

test sets of the 20-Newsgroups and IMDB datasets. We therefore obtain test F_{micro} and F_{macro} score distributions over 20 seeds for each model and dataset. Tables 3.3-3.4 list the mean F_{micro} and F_{macro} scores respectively along with standard deviations over 20 seeds for the CNN-1 and DKL-lin models and their pretrained variants. We list the results for the CNN-2 and DKL-RBF models in Appendix A as we found that adding a second dense layer to the CNN or using an RBF kernel instead of a linear kernel in the DKL model either hurt or did not change model performance.

We compared the performances of all 11 models (across Tables 3.3-3.4, 3.5, and 1-2 in Appendix A), giving 55 pairwise model comparisons for each dataset and each averaging scheme (micro vs. macro). We compared the mean scores for each pair across 20 seeds using a one-tailed paired t test at a total level of significance 0.05 and thus a per-comparison level of significance $\alpha = 0.0011$ after a Bonferroni adjustment.

The DKL-lin model (either unpretrained or pretrained with CNN) outperforms both CNN-1 baselines (both unpretrained and pretrained with CNN) on all datasets, with the performance gap statistically significant on the 20Newsgroups and IMDB datasets, thereby allowing us to conclude that DKL can be beneficial for text classification. It follows that the DKL models also outperform the CNN-2 models, telling us that the success of DKL is not just due to a more powerful classifier. Pretraining DKL-lin with a CNN boosts performance only on EPR and IMDB-1, suggesting that pretraining is sometimes beneficial on smaller datasets. Since the CNN-pretrained DKL-lin outperforms the CNN-pretrained CNN-1 model, the benefit of pretraining is more than just a longer training duration.

Table 3.5 lists both the mean F_{micro} and F_{macro} scores along with standard deviations over 20 seeds for the CNN-SVGP and DKL-LSC models, whose feature networks are held fixed during training. CNN-SVGP performs poorly on all datasets– even worse than the CNN-1 model from which the feature network was obtained. This implies that it is not sufficient to simply fit an SVGP classifier on fixed, already-trained CNN features; the effectiveness of DKL rests on learning feature representations that are compatible with an SVGP classifier and that are different from standard CNN features. Indeed, the dominance of the unpretrained DKL-lin model over the CNN-pretrained DKL-lin model on most datasets again indicates that DKL features trained from random initialization are often better than those that are modified from pretrained CNN features.

DKL-LSC often displays performance comparable to that of the best DKL model from which its feature network was obtained. This again suggests that much of the power of DKL comes not from having an SVGP classifier at test time but from the DKL features that are learned through Bayesian training; the DKL features are robust and remain useful even when transferred onto a non-Bayesian architecture. Moreover, the DKL-pretrained CNN-1 model is consistently worse than DKL-LSC, meaning that performance deteriorates as the pretrained DKL features are updated through a linear softmax classifier with non-Bayesian training.

3.5.2 Feature visualizations

Our results suggest that DKL models are often able to outperform CNN models by extracting better feature representations. We used t-SNE [103] to visualize the feature representations of the 20-Newsgroups test set extracted by various CNN and DKL models trained on 20-Newsgroups-100 (Fig. 3.2). The quality of clustering by true class label is better for models

Table 3.3: Mean test F_{micro} scores (as percentages) with standard deviations across 20 random seeds for each model and dataset. A score is in bold if the mean performance of the model is greater than that of all other models for a given dataset, at a level of significance $\alpha = 0.0011$. The best DKL-lin model (either unpretrained or pretrained with CNN) significantly outperforms the CNN-1 baselines (unpretrained and pretrained with CNN) on all datasets except EPR. Pretraining DKL-lin boosts performance on EPR and IMDB-1– the two datasets with the smallest training sets. The small performance gap between the best DKL-lin model and the CNN-1 baseline on EPR and IMDB-1 may be due to a yet unknown trainability issue due to small training sets, and the small performance gap on IMDB-100 may be because it is a class-balanced dataset in the large-sample regime.

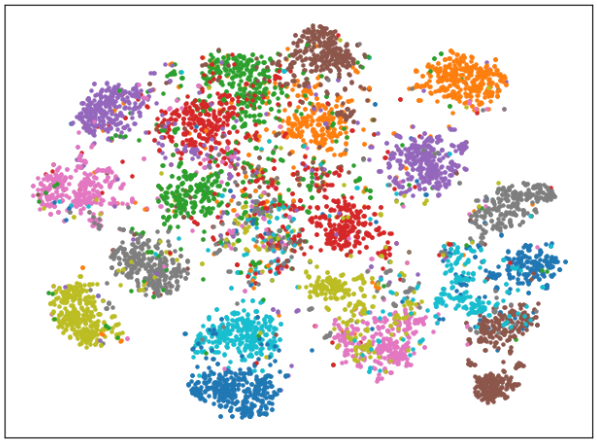
Dataset	CNN-1 (scratch)	CNN-1 (pre. CNN)	CNN-1 (pre. DKL)	DKL-lin (pre. CNN)	DKL-lin (scratch)
EPR	86.2 ± 0.6	86.5 ± 0.6	86.6 ± 0.7	86.6 ± 0.4	83.8 ± 0.7
20News-22	68.9 ± 1.0	67.9 ± 0.8	70.0 ± 0.9	70.8 ± 0.7	$\mathbf{75.7 \pm 0.8}$
IMDB-1	72.8 ± 1.7	72.1 ± 1.7	69.9 ± 1.9	73.8 ± 1.9	73.5 ± 3.2
20News-100	79.0 ± 0.4	78.7 ± 0.5	79.0 ± 0.5	82.6 ± 0.5	$\mathbf{83.4 \pm 0.5}$
IMDB-5	79.2 ± 0.8	77.6 ± 0.7	77.5 ± 0.7	77.1 ± 1.0	$\mathbf{82.9 \pm 0.5}$
IMDB-100	88.7 ± 0.3	88.3 ± 0.4	88.8 ± 0.2	88.9 ± 0.3	89.1 ± 0.3

Table 3.4: Mean test F_{macro} scores (as percentages) with standard deviations across 20 random seeds for each model and dataset. A score is in bold if the mean performance of the model is greater than that of all other models for a given dataset, at a level of significance $\alpha = 0.0011$. The F_{macro} scores on the 20-Newsgroups and IMDB datasets are approximately equal to the F_{micro} scores (Table 3.3) since the test set and test predictions are both nearly class-balanced. The F_{macro} scores on EPR are lower than the F_{micro} scores due to class imbalance but display the same trend across the models.

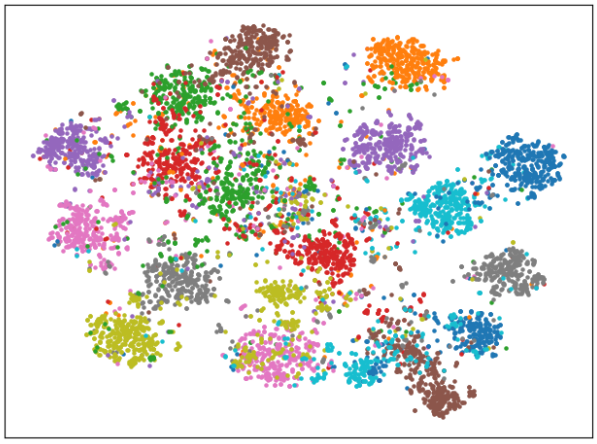
Dataset	CNN-1 (scratch)	CNN-1 (pre. CNN)	CNN-1 (pre. DKL)	DKL-lin (pre. CNN)	DKL-lin (scratch)
EPR	82.8 ± 0.9	83.4 ± 0.9	83.6 ± 1.0	83.6 ± 0.6	78.1 ± 1.3
20News-22	68.8 ± 0.9	67.8 ± 0.8	69.6 ± 0.9	70.3 ± 0.7	$\mathbf{75.0 \pm 0.8}$
IMDB-1	72.6 ± 1.8	72.0 ± 1.7	69.8 ± 1.9	73.8 ± 1.9	73.4 ± 3.3
20News-100	78.5 ± 0.4	78.3 ± 0.5	78.4 ± 0.5	82.1 ± 0.5	$\mathbf{82.8 \pm 0.5}$
IMDB-5	79.2 ± 0.8	77.6 ± 0.7	77.4 ± 0.8	77.0 ± 1.0	$\mathbf{82.9 \pm 0.5}$
IMDB-100	88.7 ± 0.3	88.3 ± 0.4	88.8 ± 0.3	88.9 ± 0.3	89.1 ± 0.3

Table 3.5: Mean test F_{micro} and F_{macro} scores (as percentages) with standard deviations across 20 random seeds for the models with fixed feature networks. A linear softmax classifier trained on DKL features (DKL-LSC) often performs as well as the best DKL-lin model (either unpretrained or pretrained with CNN, listed in Tables 3.3-3.4). A score is in bold if the mean performance of the model is not different from or is greater than that of the best DKL-lin model, at a level of significance $\alpha = 0.0011$.

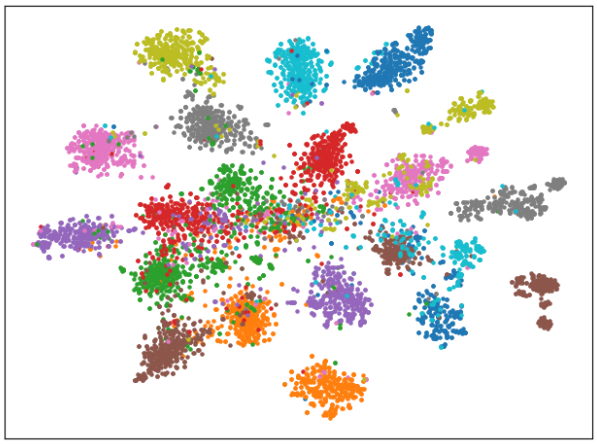
	Micro		Macro	
Dataset	CNN-SVGP	DKL-LSC	CNN-SVGP	DKL-LSC
EPR	72.0 ± 2.2	85.3 ± 0.6	61.0 ± 3.6	81.3 ± 1.0
20News-22	69.0 ± 0.9	74.8 ± 1.0	68.6 ± 0.8	74.4 ± 0.9
IMDB-1	57.0 ± 4.6	$\mathbf{73.9 \pm 2.0}$	56.9 ± 4.6	$\mathbf{73.8 \pm 2.0}$
20News-100	78.6 ± 0.7	$\mathbf{83.1 \pm 0.5}$	78.1 ± 0.7	$\mathbf{82.6 \pm 0.4}$
IMDB-5	79.0 ± 1.5	$\mathbf{83.2 \pm 0.5}$	79.0 ± 1.5	$\mathbf{83.1 \pm 0.5}$
IMDB-100	88.5 ± 0.3	$\mathbf{89.2 \pm 0.2}$	88.5 ± 0.3	$\mathbf{89.2 \pm 0.2}$



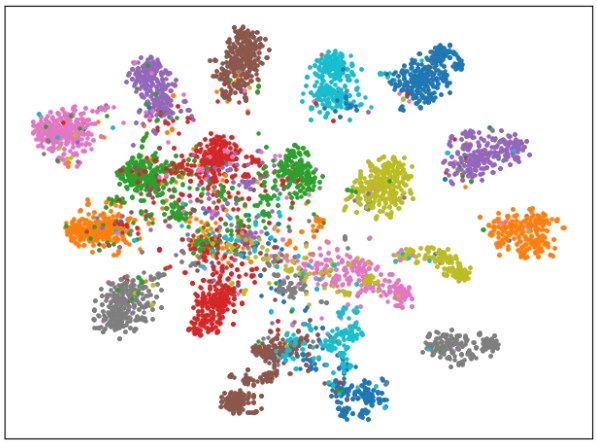
(a) CNN-1
($F_{\text{micro}} = 78.2\%$)



(b) CNN-1 (pre. DKL)
($F_{\text{micro}} = 79.0\%$)



(c) DKL-lin
($F_{\text{micro}} = 84.1\%$)



(d) DKL-lin (pre. CNN)
($F_{\text{micro}} = 83.0\%$)

Figure 3.2: *t*-SNE visualizations of the 20-Newsgroups test set passed through the feature networks of four different models trained on 20-Newsgroups-100. These models are based on only one random seed– the seed giving the biggest performance gap between the CNN-1 and DKL-lin models. Each color represents one ground truth class label. The "pure" DKL features (c) are clustered most clearly. The CNN-pretrained DKL feature clusters (d) are not as well-separated. The quality of clustering declines further if the DKL features are fine-tuned through a linear softmax classifier (b). The baseline CNN features (a) are the least well-separated.

that achieve greater classification performance. Moreover, the quality of clustering (and thus classification performance) improves with the "purity" of DKL features; pretraining the DKL features with a CNN or fine-tuning them through a linear softmax classifier deteriorates not only performance but quality of clustering as well.

3.5.3 Comparison of uncertainty quantification

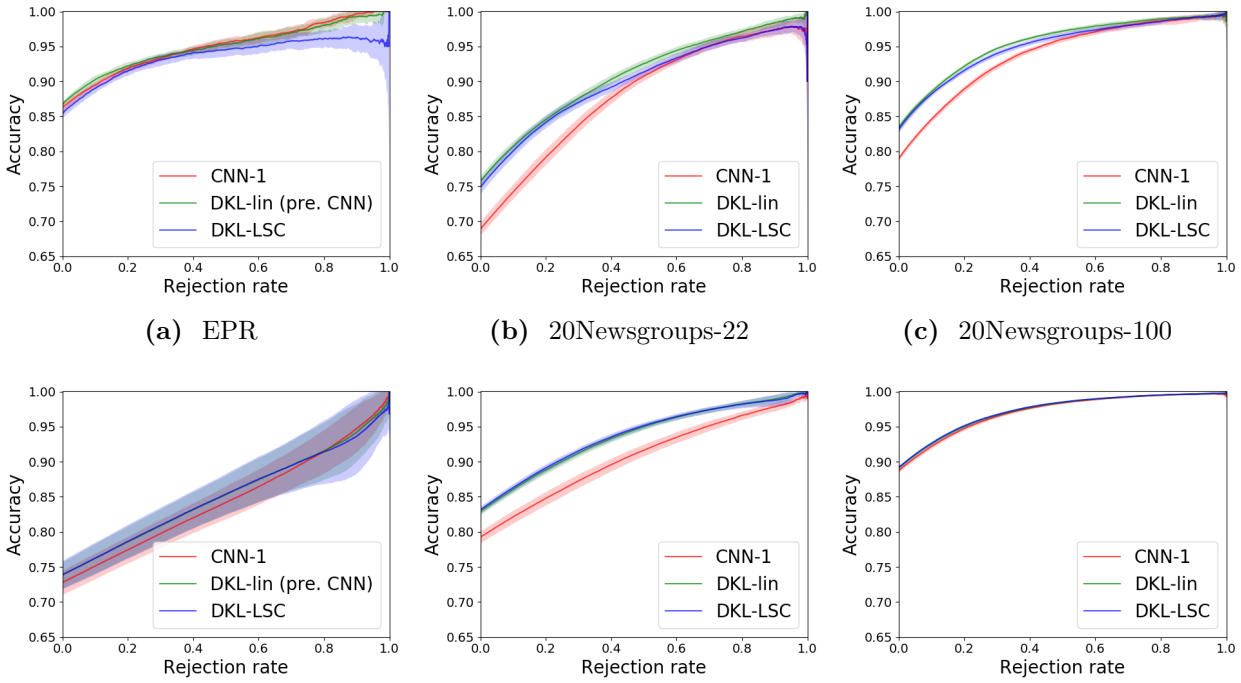
We examined the UQ capabilities of the DKL-lin and DKL-LSC models as compared to the CNN-1 baseline. We calculated an ARC on the test predictions of each model on each dataset for the 20 random seeds (for the EPR dataset, we first aggregated the predictions from all ten CV test folds). We used the probability assigned to the predicted class label as the confidence score. We averaged the ARCs over each set of 20 seeds by averaging the accuracies at each rejection rate.

On almost all datasets, the DKL-lin and DKL-LSC models maintain greater accuracies than the CNN-1 baseline (Fig. 3.3), indicating that DKL boosts performance while preserving UQ. Furthermore, if we set a target accuracy of 95% and if we find the rejection rates at which the CNN-1 and DKL-lin models attain the target accuracy based on their mean ARCs, then we see that on the moderately-sized datasets (20Newsgroups-100 and IMDB-5), the DKL-lin model rejects about 30% fewer test inputs as compared to the CNN-1 baseline– highlighting the practical utility of DKL for UQ.

3.6 Discussion

Based on our results, we conclude that DKL can often be a useful paradigm for boosting the performance of CNN models for text classification and that it can potentially be beneficial for extracting information from cancer pathology reports. Moreover, when DKL helps, it is often because it facilitates the model to learn better feature representations. We noticed high variability in test performance due to random network initialization and train-validation-test splits, with the standard deviations of the F_1 scores often on the order of 1 percentage point. This suggests that we cannot claim that a model outperforming a baseline by only a small margin is truely a better model based only on single runs. It is instead critical that we compare the distributions of test scores obtained from many trails of experiments with several random seeds.

Our results also raise several questions that would be interesting to explore as future work. First, when is it necessary to pretrain a DKL model? We found that pretraining helped to boost the performance of the DKL model only on EPR and IMDB-1 – the two datasets with the smallest training sets – although the DKL model outperforms the CNN baseline on IMDB-1 even without pretraining. Pretraining should therefore be considered when training examples are scarce. Even if data is abundant, some form of pretraining may still be needed if there is sufficient class imbalance such that one class is comparable in size to, say, IMDB-1. This may be relevant for tasks such as primary site or histology information extraction from larger EPR datasets. Alternative initialization and training strategies that can rid the need for pretraining altogether will be part of future work.



(a) EPR **(b)** 20Newsgroups-22 **(c)** 20Newsgroups-100

(d) IMDB-1 **(e)** IMDB-5 **(f)** IMDB-100

Figure 3.3: Accuracy-rejection curves (ARCs) for the best models (DKL-lin and DKL-LSC) and the CNN-1 baseline for each dataset. The accuracy at each rejection rate is an average over 20 random seeds, with the shaded area indicating the range within one standard deviation of the mean. The DKL-lin and DKL-LSC models often maintain higher ARCs than the CNN-1 baseline on almost all datasets, suggesting higher overall accuracy while preserving UQ. The accuracy gap between the DKL models and the CNN baseline is less prominent at lower rejection rates on EPR (a) and IMDB-1 (d) – which have the smallest training sets – and on IMDB-100 (f). We speculate that this is due to an unidentified trainability issue on small datasets for EPR and IMDB-1 and due to a large class-balanced training set for IMDB-100.

Second, is DKL relevant to larger datasets? Bayesian approaches are expected to do well on small datasets, where regularization is critical to avoid overfitting. The performance gap between CNN and DKL models is small on very small datasets (e.g. EPR and IMDB-1), possibly due to some trainability issues, where pretraining also remained necessary. The benefit of Bayesian training contracts when more training examples are available, as observed going from 20Newsgroups-22 to 20Newsgroups-100 and from IMDB-5 to IMDB-100. However, DKL can still be relevant on larger datasets if there is enough class imbalance such that The data belonging to rare classes fall into the low-sample regime. Such extreme class imbalance is known to exist in larger EPR datasets [6], which motivates the application of DKL for such problems.

Third, is DKL useful for pretraining traditional DL models? Our results suggest that a CNN pretrained with the feature network of a DKL model can achieve performance that often matches and even exceeds that of the DKL model itself. This suggests that DKL is beneficial not just because it allows for Bayesian inference at test time but also because it extracts robust features through Bayesian training that can be transferred to non-Bayesian models. DKL can thus be understood as a method that augments a DL loss function to derive better feature representations; once training is done, we may be able to discard DKL. This viewpoint suggests that DKL may be an entire paradigm applicable to other DL architectures such as attention networks that have achieved state-of-the-art performance on information extraction from cancer pathology reports [52, 51]. Examining the robustness or sensitivity of CNN features extracted through DKL – e.g., to adversarial attacks via small perturbations in the input – is another avenue for future research.

Finally, is DKL useful for UQ? An important feature of a DKL model is that it returns a full predictive posterior distribution, which may yield predictive class probabilities that are better calibrated than the softmax scores of a traditional CNN. We found that DKL is often able to boost the performance of a CNN model while preserving UQ, in the sense that DKL remains beneficial on subsets of increasingly more confident predictions. DKL can also often help to attain a target accuracy while significantly reducing the rejection rate. The UQ capability of DKL will be even more prominent on out-of-distribution (OOD) examples since GP models associate predictive variance with distance from the training set in feature space. OOD detection can be relevant for pathology reports containing rare or even new cancers not accounted for in training data. Additionally, in contrast to other UQ methods such as Bayesian neural networks, the current formulation of DKL does not consider uncertainty in the feature network. Incorporating variance into the DKL feature network could further improve OOD detection, since it could prevent OOD examples from being mapped into the linear span of the inducing points, where high confidence is assigned (see Sec. 3.3.4). However, UQ in the DKL feature network is computationally expensive. The trade-off between UQ in the feature network and scalability to large network architectures remains an open challenge for future work.

The purpose of this study was to empirically demonstrate that DKL can be applied to text classification tasks and that DKL can be beneficial for information extraction from pathology reports. The questions raised in this discussion serve as motivation to extend DKL to much larger EPR datasets with multiple classification tasks with extreme class imbalance. The application of DKL to large EPR datasets remains an opportunity to be examined. We will avail ourselves of this opportunity and address the associated challenges posed by the

sheer scale of the data and the characteristics of different tasks and extreme class imbalance in its entirety in a future study.

Chapter 4

Bayesian Neural Networks for Information Extraction from Cancer Pathology Reports

Disclosure

A version of this chapter has been accepted as an extended abstract at the Computational Approaches for Cancer workshop at Supercomputing, 2020. The authors on the accepted abstract are Devanshu Agrawal, Shang Gao (Oak Ridge National Laboratory), Eric Durbin (University of Kentucky), Xiao-Cheng Wu (Louisiana State University Health Sciences Center), Jennifer Doherty (University of Utah), Antoinette Stroup (Rutgers Cancer Institute), Linda Coyle (Information Management Services Inc.), Lynne Penberthy (National Cancer Institute), Georgia Tourassi (Oak Ridge National Laboratory), and Jacob Hinkle (Oak Ridge National Laboratory).

Devanshu Agrawal and Jacob Hinkle conceptualized the paper and developed the code libraries. Devanshu Agrawal implemented and ran the experiments, with support from Shang Gao regarding the deterministic HiSAN. Eric Durbin, Xiao-Cheng Wu, Jennifer Doherty, Antoinette Stroup, Linda Coyle, and Lynne Penberthy oversaw and organized the preparation of the pathology reports. All authors read and approved the manuscript.

The only revisions made since the original submission are additional figures (validation curves for the HiSAN) and formatting.

Abstract

Cancer pathology reports are a rich source of information for the national cancer surveillance effort. Deep learning is currently the state-of-the-art (SOTA) methodology to automate information extraction from cancer pathology reports (IE-CPR), where information extraction is treated as a multi-task classification problen, with each task corresponding to a key data element such as tumor site or histological type. However, on tasks with extreme class imbalance, the SOTA models perform poorly on the minority classes. Moreover, the tasks require different training times, making early stopping difficult. Hypothesizing that both challenges are due to overfitting, we propose Bayesian neural networks as a solution. In this work, we implement Bayesian versions of two SOTA models– the shallow-wide convolutional neural network (CNN) and the hierarchical self-attention network (HiSAN). We find that the Bayesian framework boosts the performances of both models, with the Bayesian CNN performing best, thus establishing a new SOTA for IE-CPR.

4.1 Introduction

National cancer surveillance is a critical activity for tracking and understanding trends in cancer incidents and mortality across the United States. Cancer pathology reports are the primary source of pertinent information for this surveillance effort, and thus a central task in this endeavor is to extract key data elements from these reports such as tumor site and histological type. Given the national scale of this activity, developing a model that can automatically extract key data elements from cancer pathology reports is imperative for a cost-effective and real-time surveillance program. However, automating information extraction from unstructured text such as pathology reports is challenging because text

corresponding to the same concept can be highly variable, such as the description of a tumor's specific location on an organ.

Deep learning (DL) is currently the state-of-the-art (SOTA) methodology for information extraction from cancer pathology reports (IE-CPR), significantly outperforming rules-based models as well as other traditional machine learning approaches [122, 52, 51, 6]. In the DL methodology, the IE-CPR problem is cast as a multi-task text classification problem where each task corresponds to a data element of interest such as tumor site or histological type. The DL model is then a composition of some multistage feature extraction pipeline and a classifier that is end-to-end differentiable and hence trainable with gradient-based optimization methods. The key to the success of DL is its ability to extract rich, generalizable feature representations from input documents.

Despite the success of the DL paradigm for the IE-CPR problem, a number of challenges remain. In this work, we focus on two related challenges: First, many tasks display extreme class imbalance; this is most prominent in the tasks with hundreds of classes– tumor site and histological type, which also happen to be the most important tasks in the IE-CPR problem. The class imbalance is reflected in the significant gap between the overall classification accuracy on a task (measured in terms of micro-averaged F_1 or "F_1-micro" score) and the unweighted average of one vs. all classification accuracies over all the classes in the task (measured in terms of macro-averaged F_1 or "F_1-macro" score); in particular, the SOTA DL models perform significantly worse on minority classes. Second, the SOTA DL models employ multi-task learning, where a single feature extraction network is shared across tasks and the tasks are trained jointly with a single objective function. Although multi-task learning has been shown to boost performance on the IE-CPR problem [6], it makes cross validation challenging since the tasks attain their optimal validation performances at different points during training time. This is exacerbated by having two metrics—F_1-micro and F_1-macro—for each task; because the loss function used by the SOTA DL models gives equal weight to all classes in a task, the F_1-micro score tends to attain its maximum value while the F_1-macro score is still rising. Consequently, the DL models tend to overfit on some tasks with respect to one metric while underfitting on others.

We suspect that at the heart of both challenges described above is the issue of overfitting and lack of robustness; in the first challenge, the models are not robust to limited data in the minority classes, and in the second challenge, overfitting of multiple tasks and metrics cannot be managed simultaneously with early stopping alone. To address both challenges, we propose the Bayesian deep learning methodology [114, 56, 19, 150]. In the Bayesian paradigm, the practitioner first selects and places a prior distribution on the model parameters and then invokes Bayes' Rule to infer a posterior distribution. Since Bayesian model predictions take into account predictions based on every parameter configuration (weighted by their consistency with observed data and proximity to the prior), these models tend to be more robust to overfitting than point-estimate models. The Bayesian methodology is particularly well-suited for DL when data is limited since DL models tend to require a large volume of data to achieve good performance. Regardless of the volume of cancer pathology reports available for training DL models, we inevitably find ourselves in the low-data regime when looking at the minority classes, and we therefore expect Bayesian DL to assist with the class imbalance problem and to help boost the F_1-macro scores at the very least. Moreover, Bayesian models typically do not overfit during training, and thus it is common in the Bayesian DL literature

to train a DL model to convergence without early stopping. We think this property can let us address the early stopping problem in multi-task learning by allowing us to train all tasks to convergence; this will let the slower-learning tasks "catch up" to the faster-learning tasks without the latter overfitting.

In this work, we implement Bayesian versions of two SOTA DL models for the IE-CPR problem– the shallow-wide convolutional neural network (CNN) [6] and the hierarchical self-attention network (HiSAN) [51]. We compare these Bayesian models to their deterministic counterparts and find that in the case of the CNN, the Bayesian framework indeed results in improved F_1-macro scores, and in the case of the HiSAN, the Bayesian framework leads to significant improvement if compared without pretrained word embeddings. Of all models tested, we find the Bayesian CNN to perform best, resulting in a new SOTA on the IE-CPR problem.

4.2 Materials and methods

4.2.1 Dataset

We used a dataset of $878,854$ cancer pathology reports gathered from four different SEER cancer registries (LA, KY, NJ, UT). The reports were manually annotated by the cancer registries using standard ICD-O-3 codes as per the SEER guidelines. We were interested in the tasks of extracting six key data elements: Behavior, grade, laterality, histology, site, and subsite[1]. Table 4.1 lists the number of classes in each task as well as the greatest-to-smallest ratio of class prevalence. We note the extreme class imbalance in the histology, site, and subsite tasks.

The reports were preprocessed from their original XML format and tokenized to integer word indices. Each report was either padded or truncated to a length of 1500 tokens.

4.2.2 Models

Convolutional neural network

Our CNN model is based on the shallow-wide architecture specifically developed for text classification [79]. This CNN represented the SOTA on the IE-CPR problem [122, 6] until the advent of the HiSAN model (see Sec. 4.2.2); nevertheless, due to its simplicity the CNN remains the primary workhorse for tackling the IE-CPR problem.

The CNN expects as its input a sequence of integer tokens, with each token indexing a word in the vocabulary. The input sequence is first passed through a word embedding layer, which maps each token to a word vector of dimension 300, which is much smaller than the vocabulary size. The embedding output is then passed through three convolutional modules in parallel with filter widths 3, 4, and 5 respectively. Each convolutional module comprises 300 convolutional filters followed by global max pooling and a rectified linear unit (reLU) activation layer. These convolutional modules therefore extract salient features localized to

[1]Descriptions of these tasks and their respective possible values (classes) can be found at https://seer.cancer.gov/tools/codingmanuals/

Table 4.1: Class count and greatest-to-smallest ratio of class prevalence for each task (data element to be extracted).

Task	Num. classes	Prev. ratio
Behavior	4	587.5
Grade	9	9763.6
Histology	572	195567.0
Laterality	7	638.2
Site	70	24872.1
Subsite	324	82177.0

3-5 consecutive tokens in the input sequence. The output vectors of all convolutional modules are then concatenated together, producing a final feature representation of dimension 900. The extracted feature representation is then finally passed through a multi-task linear softmax classifier

Hierarchical self-attention network

The HiSAN represents the current SOTA on the IE-CPR problem, achieving slightly better performance than the CNN [51]. It is based on the attention mechanism that is now ubiquitous in natural language processing [11, 163, 143]. More specifically, the HiSAN replaces the three parallel convolutional modules in the CNN with two attention modules composed together; the first attention module—applied to the output of the word embedding layer—operates at the word level, while the second attention module—applied to the output of the word-level attention module—operates at the sentence or line level, thereby facilitating hierarchical feature extraction. Because of this hierarchical architecture, all input documents are reshaped into arrays of lines and words. In this work, we simply reshape each 1500-dimensional integer-tokenized report into a 150×10-dimensional array, representing 150 lines of 10 words each.

Each attention module comprises a self-attention layer and a "target attention" layer. The self-attention layer first transforms its input by three separate linear transformations (8 attention heads each of dimension 64) to obtain "queries", "keys", and "values". The value vectors are then summed together, with the "similarity" between the key and query vectors acting as weighting coefficients. Intuitively, each unit (either a word or a line) is weighted by its relevance within the context furnished by the other units in the input. The output of the self-attention layer is then passed into the target attention layer, which weights each unit in its input by its relevance to a learnable query variable or context vector. In the final output of the word-level (resp. line-level) attention module, each line (resp. document) is represented as a 512-dimensional vector. The document embedding returned by the line-level attention module is then fed into a multi-task linear softmax classifier.

Bayesian models

The Bayesian paradigm is appealing as it offers a principled approach to improve a model's robustness to overfitting. Unfortunately, however, exact Bayesian inference on neural networks is intractable and must be approximated. We implement the Bayesian CNN and Bayesian HiSAN models using variational inference (VI)– the most popular method for approximate Bayesian inference on neural networks [56, 19]. In VI, we propose a family of variational distributions and seek the one that best approximates the posterior in the sense of minimum KL divergence. This results in a loss function that can be minimized with standard stochastic gradient descent methods. In this work, we select a fully factorized (i.e., mean field) Gaussian as the variational distribution. We take the prior distribution to be a zero-centered isotropic Gaussian. We estimate the expected log-likelihood term in the loss using Monte Carlo sampling and the reparameterization trick. We employ the local reparameterization trick on the linear layers to reduce sampling complexity [82]. For the convolutional layers, we use

the "flipout" method to reduce correlation between activations across different inputs in a minibatch [150].

Most of the parameters in both the CNN and HiSAN models are contained in the embedding layer, and thus an efficient and effective Bayesian implementation of the embedding layer is paramount. Direct sampling of the embedding layer parameters would be extremely inefficient due to the large parameter count, and in any case it is unnecessary since only a small fraction of the parameters are active for any given minibatch of inputs. Moreover, the out-of-the-box local reparameterization trick cannot be applied to the embedding layer since its output activations are not necessarily independent; if a word appears multiple times in an input document, then the sampled embedding vectors corresponding to all copies of this word will be identical. To circumvent this problem, we adapt the local reparameterization trick for the embedding layer by introducing the concept of a "local vocabulary". For each input document, we define its local vocabulary to be the set of unique words appearing in the document. We use the traditional local reparameterization trick to draw one sample of "local embeddings"– the embeddings of the local vocabulary; this works since the embedding vectors of the (distinct) words in the local vocabulary are independent. We then transform the document so that its words are represented as indices into its local vocabulary (as opposed to the global vocabulary), and we pass this representation through the "local embedding matrix" to look up the appropriate embedding samples.

4.2.3 Experimental setup

Metrics and validation

We measure the classification performance of all models with the F_1 score– the harmonic mean of precision and recall. We calculate both micro-averaged and macro-averaged F_1 scores (F_1-micro and F_1-macro respectively).

We perform ten-fold stratified cross validation (CV) and measure test metrics separately on each of the ten folds. For each of the ten training-test splits, we further split the training fold into training and validation sets, stratified to maintain relative class frequencies. We take the validation set to be 20% of the training set.

Initialization and training details

We initialize all layers except the embedding layer in the deterministic CNN and HiSAN models with Xavier initialization. We initialize the embedding layer randomly using the uniform distribution $[-0.05, 0.05]$ as we found it to work well.

For the Bayesian models, we initialize the variational means in identical fashion to their deterministic counterparts (Xavier initialization). We initialize all variational variances to a small value $\ln(1 + e^{-6})$. We set the isotropic prior variance of each layer equal to the (Xavier) initialization variance of the corresponding deterministic layer, times some scalar multiplier, whose value we determine through cross validation. Our motivation for this choice of prior is that our best belief about the network parameters before training is precisely the initialization distribution itself; we found in practice, however, that this does not always yield the best validation performance– hence the introduction of the prior variance multiplier.

We train all classification models using the Adam optimizer with learning rate 10^{-4} and minibatch size 258 until the validation loss either converges or clearly starts to increase on all CV folds (We use minibatch size 258 since it can be evenly distributed across six MPI ranks; see Sec. 4.2.3). For each fold, we take the model checkpoint from the epoch giving the lowest validation loss as the final model. We apply dropout on the convolutional modules of the deterministic CNN and after the embedding layer and each of the attention modules of the deterministic HiSAN; we use dropout rates of 0.1 and 0.5 for the CNN and HiSAN respectively, as we found these values to work best in terms of validation performance.

Software and hardware resources

We implement all models and run all experiments in Pytorch. For the Bayesian models, we developed a simple wrapper function that returns the Bayesian version of an input Pytorch model.

We run all experiments on the Summit supercomputer. To speed up training, we implement data parallelism, with each seed and fold distributed across six MPI ranks (each Summit node has six NVIDIA Tesla V100 GPUs).

4.3 Results

Tables 4.2-4.3 list the mean F_1-micro and F_1-macro scores respectively along with standard deviations over 10 CV folds for all models on all tasks. We use the prefixes "B" and "D" to mean "Bayesian" and "deterministic" respectively. In addition to the deterministic and Bayesian CNN and HiSAN models, we implement and include results for a deterministic HiSAN whose embedding layer is initialized with word2vec embeddings learned from scratch on the cancer pathology reports dataset (D-HiSAN-W2V), as this has represented the SOTA on the IE-CPR problem until this work [51]. The B-CNN achieves higher F_1-macro scores than does the D-CNN, most notably on histology and subsite– the tasks with the greatest numbers of classes and the most extreme cases of class imbalance. The performance gap between the B-HiSAN and D-HiSAN is even more striking, with the former achieving better performance across the board. However, the D-HiSAN-W2V remains the best-performing HiSAN model, suggesting that a strong initialization of the embedding layer is more beneficial than the Bayesian framework alone. Overall, the B-CNN is the best-performing model and even outperforms the D-HiSAN-W2V, thus establishing a new SOTA on the IE-CPR problem.

The validation curves of the CNN and HiSAN (without word2vec) models on the histology and subsite tasks (Fig. 4.1) illustrate how the Bayesian models are able to outperform the deterministic ones. For the deterministic models, we observe that: 1) The D-CNN and D-HiSAN eventually overfit on both tasks; 2) The F_1-micro scores attain their maxima while the F_1-macro scores are still increasing; and 3) the F_1-micro scores attain their maxima at different epochs on the two tasks. In contrast, the validation curves of the B-CNN and B-HiSAN appear to converge and thus allow for all metrics to attain their optima simultaneously at some sufficiently large number of epochs; in particular, this allows the F_1-macro scores to continue increasing without the F_1-micro scores deteriorating.

Table 4.2: Mean test F_1-micro scores (as percentages) with standard deviation for each model.

Task	D-CNN	B-CNN	D-HiSAN	B-HiSAN	D-HiSAN-W2V
Behavior	97.6 ± 0.3	97.6 ± 0.2	96.6 ± 0.3	97.4 ± 0.2	97.4 ± 0.2
Grade	74.3 ± 1.8	74.8 ± 2.1	71.7 ± 1.9	73.8 ± 2.1	74.0 ± 2.0
Histology	77.0 ± 1.0	77.6 ± 1.2	74.0 ± 1.3	76.7 ± 1.3	77.0 ± 1.2
Laterality	90.9 ± 0.7	90.9 ± 0.4	90.4 ± 0.5	90.8 ± 0.5	91.0 ± 0.5
Site	92.3 ± 0.5	92.5 ± 0.6	91.5 ± 0.6	92.3 ± 0.5	92.5 ± 0.6
Subsite	64.2 ± 1.1	66.4 ± 1.3	62.8 ± 1.3	65.3 ± 1.1	65.4 ± 1.2

Table 4.3: Mean test F_1-macro scores (as percentages) with standard deviation for each model.

Task	D-CNN	B-CNN	D-HiSAN	B-HiSAN	D-HiSAN-W2V
Behavior	89.6 ± 1.7	89.3 ± 1.5	83.2 ± 1.7	87.3 ± 1.2	88.3 ± 1.2
Grade	65.8 ± 3.2	67.8 ± 5.4	58.9 ± 5.1	65.1 ± 6.1	66.2 ± 5.1
Histology	31.8 ± 0.5	38.4 ± 1.2	24.4 ± 1.5	33.6 ± 1.7	34.8 ± 1.7
Laterality	51.1 ± 2.2	52.8 ± 1.4	48.2 ± 1.5	51.2 ± 1.4	50.8 ± 1.6
Site	66.7 ± 0.8	68.8 ± 1.0	63.0 ± 0.8	68.2 ± 0.8	68.8 ± 0.8
Subsite	29.6 ± 0.3	33.3 ± 0.7	26.0 ± 0.7	31.6 ± 0.5	32.6 ± 0.4

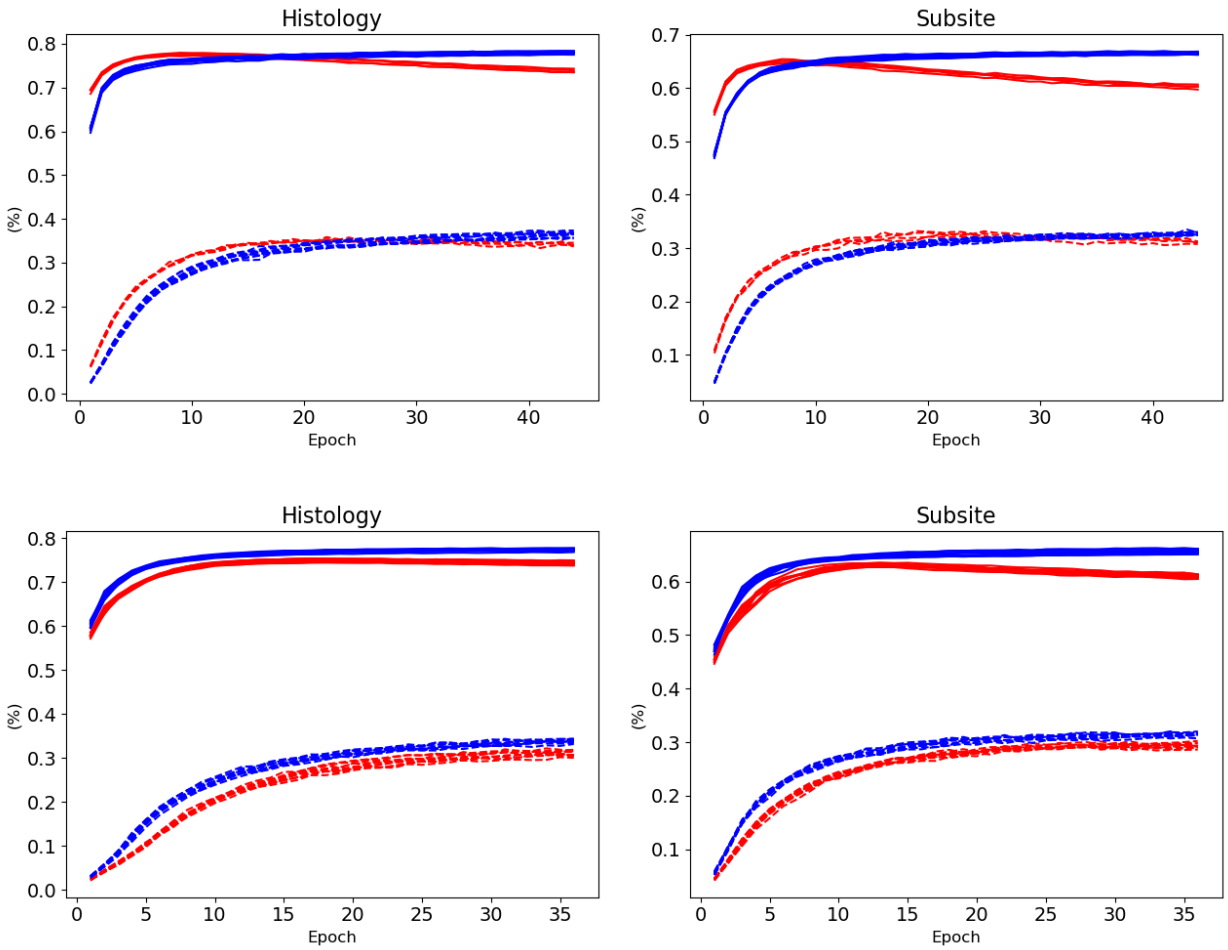


Figure 4.1: Validation F_1-micro (solid) and F_1-macro (dashed) scores of the deterministic (red) and Bayesian (blue) CNN (top) and HiSAN (bottom) models without word2vec on the histology (left) and subsite (right) tasks during training.

4.4 Discussion

Based on our results, we conclude that applying the Bayesian methodology to existing DL models can significantly boost their performance on the IE-CPR problem for the reasons surmised in Sec. 4.1. When making predictions with the Bayesian models at test time, we use their most probable parameter values (i.e., the variational means), so that the Bayesian models operate as deterministic models. This allows us to see the Bayesian methodology as a regularization framework applied during training; in this light, our results reveal that good regularization is as important as the network architecture. Indeed, the B-CNN outperforms even the D-HiSAN-W2V– establishing a new SOTA on the IE-CPR problem and suggesting that the key was to enhance the trainability of the CNN– not its expressivity.

As part of ongoing work, we are developing a Bayesian HiSAN that supports a word2vec-initialized embedding layer (B-HiSAN-W2V), since we think such a model could outperform both the B-HiSAN and D-HiSAN-W2V models. Moreover, given that the D-HiSAN-W2V outperforms the D-CNN, we hypothesize that the B-HiSAN-W2V may similarly outperform even the B-CNN model. To this end, we are developing and testing a Bayesian word2vec model, so that we may transfer the entire variational distribution of the word2vec model to the Bayesian HiSAN embedding layer. We suspect that the choice of an appropriate prior will be critical for the success of the B-HiSAN-W2V– e.g., setting the prior of the embedding layer in terms of the Bayesian word2vec variational distribution. Indeed, in the experiments presented here, we found that setting the prior variance to be 100 times the Xavier initialization variance worked well; we hope to explain this empirical finding in future work.

As other avenues for future work, we plan to investigate more sophisticated Bayesian DL methodologies, such as those that model parameter covariance [100] or initialize the prior with empirical Bayes as in the recently introduced MOPED method [83]. Finally, we plan to explore methods to control variance in Bayesian DL models using hyperpriors, with the aim to further boost the performance of these Bayesian models.

Chapter 5

Deep Ensemble Kernel Learning

Disclosure

A version of this chapter has been submitted to the International Conference on Learning Representations, 2021. The authors of the submission are Devanshu Agrawal and Jacob Hinkle.

Devanshu Agrawal and Jacob Hinkle conceptualized the paper and developed the experiments. Devanshu Agrawal implemented and ran the experiments. Devanshu Agrawal wrote the manuscript with support from Jacob Hinkle. Both authors read and approved the manuscript.

The only revisions made to this chapter since the original submission are formatting and renaming of the last section.

Abstract

Gaussian processes (GPs) are nonparametric Bayesian models that are both flexible and robust to overfitting. One of the main challenges of GP methods is selecting the kernel. In the deep kernel learning (DKL) paradigm, a deep neural network or "feature network" is used to map inputs into a latent feature space, where a GP with a "base kernel" acts; the resulting model is then trained in an end-to-end fashion. In this work, we introduce the "deep ensemble kernel learning" (DEKL) model, which is a special case of DKL. In DEKL, a linear base kernel is used, enabling exact optimization of the base kernel hyperparameters and a scalable inference method that does not require approximation by inducing points. We also represent the feature network as a concatenation of an ensemble of learner networks with a common architecture, allowing for easy model parallelism. We show that DEKL is able to approximate any kernel if the number of learners in the ensemble is arbitrarily large. Comparing the DEKL model to DKL and deep ensemble (DE) baselines on both synthetic and real-world regression tasks, we find that DEKL often outperforms both baselines in terms of predictive performance and that the DEKL learners tend to be more diverse (i.e., less correlated with one another) compared to the DE learners.

5.1 Introduction

In recent years, there has been a growing interest in Bayesian deep learning (DL), where the point predictions of traditional deep neural network (DNN) models are replaced with full predictive distributions using Bayes' Rule [114, 153]. The advantages of Bayesian DL over traditional DL are numerous and include greater robustness to overfitting and better calibrated uncertainty quantification [58, 77]. Furthermore, the success of traditional DL already rests on a number of probabilistic elements such as stochastic gradient descent (SGD), dropout, and weight initialization– all of which have been given Bayesian interpretations [133, 49, 82, 130, 75], so that insights into Bayesian DL may help to advance DL as a whole.

Gaussian processes (GPs) are nonparametric Bayesian models with appealing properties, as they admit exact inference for regression and allow for a natural functional perspective suitable for predictive modeling [124]. While at first glance GPs appear unrelated to DL models, a number of interesting connections between GPs and DNNs exist in the literature,

suggesting that GPs can constitute a valid approach to Bayesian DL [113, 90, 40, 75, 36, 129, 1]. A GP prior is typically characterized by its covariance function or "kernel", which determines the class of functions that the GP can model, as well as its generalization properties outside training data. Kernel selection is the primary problem in GP modeling, and unfortunately traditional kernels such as the radial basis function (RBF) kernel are not sufficiently expressive for complex problems where more flexible models such as DNNs generally perform well. This is the key motivation for kernel learning, which refers to the selection of an optimal kernel out of a family of kernels in a data-driven way.

A number of approaches to kernel learning exist in the literature, including some that parameterize kernels using DNNs [167, 94, 24, 132]. However, these approaches are not specific to GPs and do not take advantage of a robust Bayesian framework. In contrast, the deep kernel learning (DKL) paradigm does exactly this; In DKL, a DNN is used as a feature extractor that maps data inputs into a latent feature space, where GP inference with some "base kernel" is then performed [154, 155, 76, 3, 21, 74, 157]. The resulting model is then trained end-to-end using standard gradient-based optimization, usually in a variational framework. We note that the DKL model is just a GP with a highly flexible kernel parameterized by a DNN. By optimizing all hyperparameters (including the DNN weights) with type II maximum likelihood estimation, the DKL model is able to learn an optimal kernel in a manner directly informed by the data, while also taking advantage of the robustness granted by the Bayesian framework. A special case of DKL that is worthy of note was considered in Dasgupta et al. [38], who use a linear base kernel and impose a soft orthogonality constraint to learn the eigenfunctions of a kernel. Although similar in spirit to the approach in this paper, their method does not make use of an efficient variational method, nor is distributed training made possible since all of the basis functions are derived from the same feature network.

In this work, we introduce the "deep ensemble kernel learning" (DEKL) model– a simpler and more efficient special case of DKL with two specifications– the base kernel is linear, and the feature network is partitioned into an "ensemble" of "learners" with common network architecture. In contrast to nonlinear kernels, the linear kernel allows us to derive an efficient training and inference method for DEKL that circumvents the inducing points approximation commonly used in traditional DKL. The hyperparameters of the linear kernel can also be optimized in closed form, allowing us to simplify the loss function considerably. Convenience aside, we show that DEKL remains highly expressive, proving that it is universal in the sense that it can approximate any continuous kernel so long as its feature network is arbitrarily wide. In other words, we may keep the base kernel simple if we are willing to let the feature network be more complex. The second specification of DEKL lets us handle the complexity of the feature network; because the feature network is partitioned, it admits easy model parallelism, where the learners in the ensemble are distributed. Moreover, our universality result only requires the number of learners to be arbitrarily large; the learners themselves need not grow (meaning fixed-capacity learners are sufficient), avoiding additional model parallelism.

From a different perspective, DEKL may be regarded as an extension of traditional ensembling methods for DNNs and in particular the deep ensemble (DE) model of Lakshminarayanan et al. [87], which is also highly parallelizable. In a DE, each DNN learner parameterizes a distribution over the variates (e.g., the mean and variance of a Gaussian

in regression, or the logits of a softmax vector in classification). Each learner is trained independently with maximum likelihood estimation, and the final predictive distribution of the DE is then defined to be a uniform mixture of the individual learner predictive distributions. Although not Bayesian itself, the DE model boasts impressive predictive performance and was shown to outperform Bayesian methods such as probabilistic back propagation [68] and MC-dropout [49]. In contrast, in DEKL, the learners are trained jointly via a shared linear GP layer. We surmise that this may help to promote diversity (i.e., low correlation) among the learners by facilitating coordination, which we verify experimentally. Unlike non-Bayesian joint ensemble training methods such as that of Webb et al. [148], we hypothesize that the DEKL learners might learn to diversify in order to better approximate the posterior covariance– an inherently Bayesian feature. We therefore expect DEKL to be more efficient than DKL and more robust than DE, by drawing on the strengths of both (see Fig. 5.1 for a comparison of model architectures).

5.2 Deep ensemble kernel learning

A DKL model is a GP whose kernel encapsulates a DNN for feature extraction [154, 155]. A deep kernel is defined as

$$K_{\text{deep}}(x_1, x_2; \theta, \gamma) = K(\varphi(x_1; \theta), \varphi(x_2; \theta); \gamma),$$

where $\varphi(\cdot; \theta)$ is a DNN with weight parameters θ and $K(\cdot, \cdot; \gamma)$ is any chosen kernel—called the "base kernel"—with hyperparameters γ. Note that the kernel hyperparameters of K_{deep} include all hyperparameters γ of the base kernel K as well as the DNN weight parameters θ. Given the expressive power of DNNs, the deep kernel is also highly expressive and may be viewed as a method to automatically select a GP model.

Our proposed method, DEKL, is a special case of DKL. Whereas RBF and Matern kernels are typically used as base kernels in DKL, in DEKL we take the base kernel to be the linear kernel:

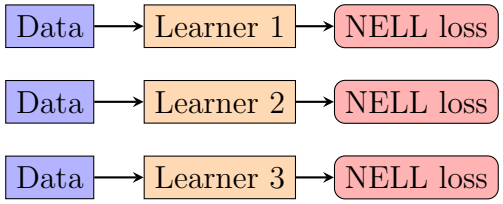
$$K_{\text{lin}}(x_1, x_2; V) = x_1^\top V x_2,$$

where V is a symmetric positive-semidefinite matrix. In order to enable parallel computation, in DEKL, we use a feature network $\varphi(\cdot; \theta)$ that is partitioned into an "ensemble" of subnetworks $\varphi_i(\cdot; \theta_i)$ called "learners", having identical network architectures; i.e., $\varphi(\cdot; \theta)$ is the concatenation of the outputs of the $\varphi_i(\cdot; \theta_i)$. DEKL offers a number of advantages over general DKL:

1. Unlike the hyperparameters of the RBF kernel, in DEKL, we can optimize the hyperparameters of the linear base kernel in closed form.

2. The linear base kernel allows us to think of a DEKL model not just as a GP but as a finite-dimensional Bayesian linear model (BLM), conditional on the feature networks; this lets us derive an efficient inference method that is much simpler than the inducing points method used in general DKL.

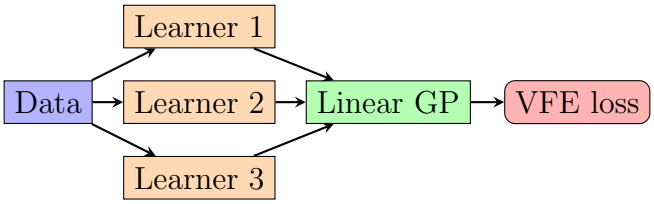
3. Finally, the partitioned architecture of the feature network makes it much more amenable to model parallelism.



(a) Deep Ensembles (DE)



(b) Deep Kernel Learning (DKL)



(c) Deep Ensemble Kernel Learning (DEKL)

Figure 5.1: Training various neural architectures considered in this work. Deep ensembles (a) use identical prediction networks with independent initialization and training. Deep kernel learning (b) uses a neural feature extraction network with an expressive kernel such as an RBF. Our model, deep ensemble kernel learning (c) uses an ensemble of feature networks trained jointly through a linear kernel that enables exact inference via minimization of variational free energy for regression.

A potential drawback to partitioning the feature network as we do in DEKL is that, compared to general DKL, the network is less expressive. However, the following universal approximation theorem for DKL implies that this effect can be compensated by adding parallel learners:

Theorem 5.1 (Universal kernel approximation theorem)**.** *Let $\mathcal{X} \subset \mathbb{R}^D$ be some compact Euclidean domain, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-polynomial activation function [119]. Then, given a continuous, symmetric, positive-definite kernel $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and any $\varepsilon > 0$, there exist a finite number H of affine functions $\beta_i : \mathcal{X} \xrightarrow{aff.} \mathbb{R}$ and a symmetric positive semi-definite matrix $V \in \mathbb{R}^{H \times H}$ such that for any $x_1, x_2 \in \mathcal{X}$,*

$$\left| \sum_{i,j=1}^{H} v_{ij} \sigma(\beta_i(x_1)) \sigma(\beta_j(x_2)) - \kappa(x_1, x_2) \right| < \varepsilon. \tag{5.1}$$

The proof, found in Appendix B, contains a straightforward combination of Mercer's Theorem [108] and the Universal Approximation Theorem, attributed to Cybenko, Hornik, Leshno, and Pinkus [34, 72, 93, 119]. Although for clarity only scalar kernels are considered in this work, multivariate versions of Thm. 5.1 are easily obtained with minor modifications to the proof. Note that the approximation in Thm. 5.1 requires a possibly large number H of affine functions β_i, but that in DEKL we replace the functions $x \to \sigma(\beta_i(x))$ with a distributed collection of strictly more flexible deep neural networks.

Theorem 5.1 defines a meaningful notion of universality that is distinct from and should not be confused with the existing notion of a "universal kernel". A kernel is called universal if it can be used to approximate under the sup-norm any continuous function as a kernel-weighted sum of values assigned at points in the domain [110]. In contrast, Thm. 5.1 defines *universality of a class of kernels*, $\mathcal{K}$, to mean that $\mathcal{K}$ is a dense set in the space of all continuous kernels (including those that are universal kernels), under the sup-norm on a compact domain.

Drawing an analogy with deep ensembles [87], we refer to the H neural networks $\varphi(\cdot; \theta_i)$ for $i = 1, \ldots, H$ as "learners" and collectively as an "ensemble". The learners $\varphi(\cdot; \theta_i)$ differ from the simpler functions $x \to \sigma(\beta_i(x))$ in Thm. 5.1 in two key respects: First, the learners $\varphi(\cdot; \theta_i)$ may be significantly more flexible than the simple learners, which can help to reduce the number H of learners required to approximate a target kernel within a given error ϵ. In the proof of Thm. 5.1, we approximate each eigenfunction of the target kernel κ with a linear combination of the learners; if the learners are sufficiently expressive, then it may take only one learner per eigenfunction to approximate the target kernel. Second, each learner $\varphi(\cdot; \theta_i)$ may have multiple outputs M. In the case of the simple learners $x \to \sigma(\beta_i(x))$, a learner with M outputs is simply a concatenation of M single-output learners, suggesting that multi-output learners may help to further reduce the number H of required learners.

5.2.1 Exact inference

Consider a dataset $\{(x_i, y_i) \in \mathbb{R}^D \times \mathbb{R}\}_{i=1}^N$. In DEKL, we model this data with likelihood $p(y \mid f)$ and GP prior $\mathcal{GP}(0, \mathcal{K}_{\text{DE}}(\cdot, \cdot; \theta, V))$, where the kernel $\mathcal{K}_{\text{DE}}(\cdot, \cdot; \theta, V) : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ is defined as

$$\mathcal{K}_{\text{DE}}(x_1, x_2; \theta, V) = \Phi(x_1; \theta)^\top V \Phi(x_2; \theta), \tag{5.2}$$

where V is an $H \times H$ symmetric positive semidefinite matrix and $\Phi(x;\theta) = \{\varphi(x;\theta_i\}_{i=1}^H$ is the concatenation of H DNN learners $\varphi(\cdot;\theta_i) : \mathbb{R}^D \mapsto \mathbb{R}$ with common network architecture. The DEKL model is thus a standard GP but with a special kernel, and thus one may proceed following standard practices for GP inference. However, in contrast to other popular GP kernels such as the RBF kernel, the kernel in Eq. (5.2) has rank at most H, and we can leverage this property to derive a more efficient GP inference method by regarding it as a Bayesian linear model (BLM) where the learners act as the BLM basis functions. More precisely, our GP prior is equivalent to taking the latent function $f : \mathbb{R}^D \mapsto \mathbb{R}$ to be a generative model of the form

$$f(x) = \Phi(x;\theta)^\top a, \quad a \sim \mathcal{N}(0, V).$$

Now formulated as a BLM, we may perform Bayesian inference by inferring the posterior on a; when the number of training points N is large, this approach is significantly more efficient than direct GP inference, which requires a Cholesky decomposition of complexity $O(N^3)$.

In the case of regression with a Gaussian likelihood, the posterior on a admits a closed form. Moreover, the evidence of the above model given the training data also admits a closed form, allowing us to obtain maximum evidence estimates of the DEKL hyperparameters V and θ as is standard practice in GP methods. However, the dependence of the evidence on θ is in general complex, so that the maximum evidence estimates must be approximated. Moreover, the evidence is not separable across the training data, so that optimization is not amenable to SGD with minibatching and is thus not scalable to large datasets. Finally, in the case of non-conjugate likelihoods such as in classification problems, exact inference on A is intractable as well. We therefore perform approximate inference and maximum evidence estimation in a variational framework, discussed in the next section. We reiterate that DEKL admits exact inference on small datasets and that maximum evidence estimation can be performed through non-stochastic gradient-based optimization methods; however, we consider the variational framework as it can handle a larger class of problems.

5.2.2 Variational inference

In variational inference, we approximate the posterior distribution on the vector a with a variational distribution by solving an optimization problem. In the GP literature, it is standard to take the variational distribution to itself be a GP. In our DEKL model formulated as a BLM, this corresponds to a normal variational distribution $\mathcal{N}(\mu, \Sigma)$ on a, where $\mu \in \mathbb{R}^H$ and $\Sigma \in \mathbb{R}^{H\times H}$. We now minimize the KL divergence between the variational and posterior distributions on a, which is equivalent to minimizing the variational free energy loss function:

$$\mathcal{L}_{\mathrm{VFE}}(\mu, \Sigma, \theta, V) = \sum_{i=1}^N \mathrm{NELL}_i(\mu, \Sigma, \theta) + \mathrm{KL}(\mu, \Sigma, V) \tag{5.3}$$

$$\mathrm{NELL}_i(\mu, \Sigma, \theta) = -\mathbb{E}_{a_i\sim\mathcal{N}(\mu,\Sigma)}\left[\log p(y_i \mid \Phi(x_i;\theta)^\top a_i)\right] \tag{5.4}$$

$$\begin{aligned}\mathrm{KL}(\mu, \Sigma, V) &= \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\mu, \Sigma) \mid \mathcal{N}(0, V)) \\ &= \frac{1}{2}\mu^\top V^{-1}\mu + \frac{1}{2}\operatorname{tr}(\Sigma V^{-1}) - \frac{1}{2}\log\det(\Sigma V^{-1}) - \frac{H}{2}.\end{aligned} \tag{5.5}$$

Observe that we optimize both the variational parameters μ and Σ as well as the kernel hyperparameters θ and V as is standard in GP inference. Note also that this loss function is separable over the training data and is thus amenable to optimization with SGD and minibatching; using a minibatch size B, we define the loss on one minibatch as the variational free energy over the batch normalized by the minibatch size:

$$\mathcal{L}_{\text{batch}}(\mu, \Sigma, \theta, V) = \frac{1}{B} \sum_{i=1}^{B} \text{NELL}_i(\mu, \Sigma, \theta) + \frac{1}{N} \text{KL}(\mu, \Sigma, V).$$

In this paper, we focus on regression tasks with Gaussian likelihood $p(y \mid f) = \mathcal{N}(y; f, \tau^{-1})$ with precision τ. The corresponding NELL admits a closed form and is given by

$$\text{NELL}_i(\mu, \Sigma, \theta) = \frac{\tau}{2} \|\Phi(x_i; \theta)^\top \mu - y_i\|^2 + \frac{\tau}{2} \Phi(x_i; \theta)^\top \Sigma \Phi(x_i; \theta) - \frac{1}{2} \log \tau + \frac{1}{2} \log(2\pi). \quad (5.6)$$

Note that in practice, we parameterize Σ with its positive-definite lower-triangular Cholesky factor in order to guarantee the symmetric positive-definiteness of Σ.

For scalability to large training sets, DKL inference is usually performed using the inducing points approximation in a sparse variational framework [139, 64]. In the case of DEKL, however, due to using a linear base kernel, the inducing points approximation is not necessary, as seen above. Instead, our parameters μ and Σ take the place of the inducing parameters in the sparse variational GP framework, and we are able to maintain an exact model for the posterior GP.

5.2.3 Optimal prior covariance

The variational free energy depends on the prior covariance V only through the KL term, which has a tractable form. This allows us to optimize V in closed form; we find the optimal prior covariance to be

$$V_*(\mu, \Sigma) = \Sigma + \mu\mu^\top. \quad (5.7)$$

Next, we substitute this optimal prior covariance back into the KL term, thereby eliminating the prior covariance from the variational free energy altogether. After some simplification, the KL term takes the following form:

$$\text{KL}_O(\mu, \Sigma) = \text{KL}(\mu, \Sigma, V_*(\mu, \Sigma)) = \frac{1}{2} \log(1 + \mu^\top \Sigma^{-1} \mu). \quad (5.8)$$

(See Appendix C for derivations of Eqs. 5.7-5.8). During training, this term drives down the signal-to-noise ratio (SNR) $\mu\Sigma^{-1}\mu$, encouraging the model to learn a more robust fit to the data.

We contrast the form of Eq. (5.8) with the KL term obtained by fixing the prior covariance to the identity $V = I$:

$$\text{KL}_I(\mu, \Sigma) = \text{KL}(\mu, \Sigma, I) = \frac{1}{2} \|\mu\|^2 + \frac{1}{2} \text{tr}(\Sigma) - \frac{1}{2} \log \det(\Sigma) - \frac{H}{2}. \quad (5.9)$$

We note that while Eq. (5.9) includes a quadratic penalty on μ, Eq. (5.8) includes only a logarithmic penalty; optimizing the prior covariance therefore results in weaker regularization

of μ. The same is true with respect to Σ; both Eqs. 5.8-5.9 include roughly logarithmic penalties on Σ^{-1} that encourage larger Σ, but Eq. (5.9) additionally includes a linear penalty on Σ, resulting in stronger regularization of Σ. We note that in regression, the NELL term already favors small Σ (see Eq. (5.6)), so that the KL term is the only thing preventing Σ from degenerating. By weakening the regularization on Σ, we suspect the KL term with optimal prior covariance to be beneficial in practice.

5.3 Experiments

In our experiments, we evaluate DEKL on various regression datasets, comparing it to two key baselines– DE and DKL (with linear base kernel). For both DKL and DEKL, we also compare the effect of the KL term with optimal prior covariance, KL_O (see Eq. (5.8)), to that of an identity prior covariance, KL_I (see Eq. (5.9)).

5.3.1 Synthetic data

For our first experiment, we consider a simple regression problem with one-dimensional inputs and outputs, as it allows for easy visualization of the posterior predictive distribution. For this purpose, we generate data via $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 3^2)$ as in Lakshminarayanan et al. [87]. We sample twenty (x, y) pairs with $x \sim \mathrm{Unif}(-4, 4)$. As already stated, we compare the DEKL model to the DKL and DE baselines. For DE, we follow the implementation of Lakshminarayanan et al. [87], except that we do not use adversarial samples, which the authors say is optional. For the DEKL and DE models, we use either one learner ($H = 1$) or five learners ($H = 5$), where each learner is a shallow MLP with rectified linear unit (ReLU) activation and 50 hidden neurons. For the DKL model, we use a shallow MLP feature network with ReLU activation and 250 hidden and five output neurons, so that it has the same total hidden width as the five-learner DE and DEKL models. Note that the only difference between the file-learner DEKL and DKL models is that—compared to the DKL model—the DEKL model is missing some connections in its second layer.

We train all models on the training set using the Adam optimizer with learning rate 10^{-3} and full batches for 1000 epochs. For the DEKL and DKL models, we set the noise precision to its true value $\tau = \frac{1}{9}$.

The predictive distributions of all models on their input domain are visualized in Fig. 5.2. We make a number of observations: First, the DKL and DEKL models with fixed prior display better uncertainty quantification than DE, having lower variance within the training domain (interpolation) and higher variance outside the training domain (extrapolation). Second, DE and DEKL with one learner have less flexible means compared to their five-learner counterparts, which we expect due to reduced expressivity. However, DE is more confident with one learner while DEKL is less confident. This highlights the difference in the uncertainty quantification mechanisms of the two models; while DE relies on multiple learners for its predictive variance, DEKL offers some predictive uncertainty even with one learner thanks to Bayesian inference. Third, using the optimal prior covariance results in overconfidence. We suspect this is because theKL term in Eq. (5.8) imposes weaker regularization on μ than does the KL term in Eq. (5.9), which may be critical on a dataset of only 20 training

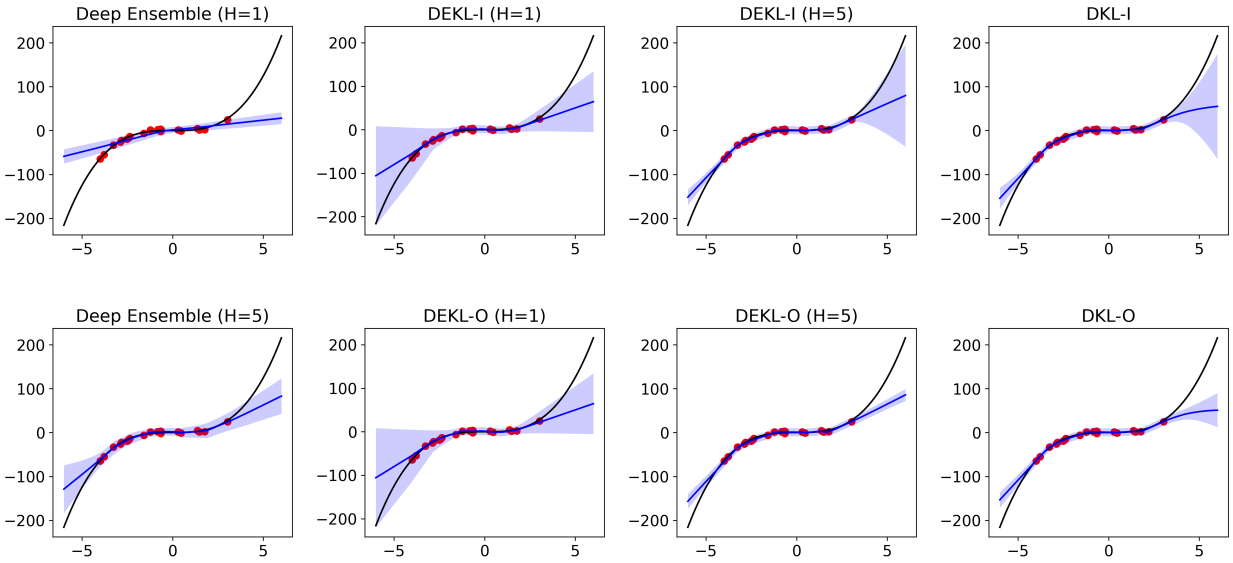


Figure 5.2: Synthetic cubic dataset. The first column shows a single DE learner and a DE model of five learners. The second and third columns show one DEKL learner and a DEKL model of five learners, both with and without optimal prior covariance. The fourth column shows a DKL model of the same width as the five-learner DEKL, both with and without optimal prior covariance. The predictive mean function (blue) with three units of standard deviation is shown for each model trained on the cubic dataset (red), with ground truth (black). Although the five-learner DEKL has far fewer parameters than DKL due to partitioning of the feature network, it approximates the full DKL well on this dataset, and improves over the equivalent DE model.

points. Finally, the five-learner DEKL predictive distribution is very similar to that of DKL, indicating that the partitioning of the feature network in DEKL grants efficiency without hurting predictive performance.

5.3.2 Real-World data

We test our DEKL model on a set of regression tasks from the UCI repository that has become a popular benchmark in the Bayesian DL literature, starting with the work of Hernández-Lobato and Adams [68]. We again compare DEKL to the DE and DKL baselines as in Sec. 5.3.1, and we use the same experimental setup as in Lakshminarayanan et al. [87]. We only consider DE and DEKL with five learners, with each learner having 50 hidden neurons, except on the Protein dataset, where we use 100 hidden neurons (and correspondingly, 500 hidden neurons in the DKL models).

We consider 20 random train-test splits of each dataset except Protein, where we use only 5 train-test splits. For each split, we use 20% of the training set for validation. We train all models on the training sets using the Adam optimizer with learning rate 10^{-3} and minibatch size 128 for 1000 epochs. For the DEKL and DKL models, we set the noise precision τ by performing a grid search and selecting the value that results in the lowest negative log-likelihood (NLL) on the validation set. For all models, we find the epoch where NLL is minimized on the validation set, then retrain all models on the full dataset up to that number of epochs, and finally evaluate the models on the test sets.

We report the test root mean square error (RMSE) and test NLL averaged over all train-test splits, as is standard for the UCI benchmarks in Bayesian DL (Tables 5.1-5.2). Given a test set with N_* data points, if y_i is the true output of the ith point and μ_i and σ_i^2 are a model's predictive mean and variance (including the noise), then the RMSE and NLL of the model are given by

$$\text{RMSE} = \sqrt{\frac{1}{N_*}\sum_{i=1}^{N_*}(y_i - \mu_i)^2}, \qquad \text{NLL} = \frac{1}{N_*}\sum_{i=1}^{N_*}\left[\frac{(y_i-\mu_i)^2}{2\sigma_i^2} + \frac{1}{2}\log\sigma_i^2\right] + \frac{1}{2}\log(2\pi),$$

where we note that the NLL is normalized by the number of data points. We see that DEKL often outperforms DE, especially in terms of RMSE, suggesting that joint ensemble Bayesian training can often be beneficial. A further comparison of DE and DEKL may be found in Appendix D, where we examine predictive performance as a function of number of learners and find that the performance gap between DE and DEKL is largely independent of the number of learners, at least up to five learners. More surprisingly, DEKL often outperforms DKL, indicating that the benefit of DEKL over DKL goes beyond simple efficiency or ease of parallelization; partitioning the feature network can often lead to a significant boost in predictive performance as well.

5.3.3 Diversity of learners

To see if joint ensemble training in a Bayesian framework promotes diversity among the learners, we compare the diversity of learners in DEKL to that in DE on the UCI datasets. Given an ensemble of learners, we measure its diversity in terms of the functional correlation

Table 5.1: Test-RMSE (mean and std. dev. across 20 train-test splits) on UCI datasets, using deep ensembles (DE), deep kernel learning (DKL) and deep ensemble kernel learning (DEKL). Suffices -I and -O indicate using KL_I (Eq. (5.9)) and KL_O (Eq. (5.8)), respectively.

Dataset	DE	DKL-I	DKL-O	DEKL-I	DEKL-O
Boston housing	3.29 ± 1.01	3.13 ± 0.77	3.43 ± 1.68	3.03 ± 0.87	3.03 ± 0.86
Concrete	5.79 ± 0.82	4.93 ± 0.80	4.79 ± 0.67	4.52 ± 0.59	4.57 ± 0.64
Energy	2.04 ± 0.31	0.61 ± 0.22	0.94 ± 0.96	0.48 ± 0.06	0.47 ± 0.05
Kin8nm	0.08 ± 0.00	0.08 ± 0.00	0.08 ± 0.01	0.07 ± 0.00	0.07 ± 0.00
Naval Propulsion	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Power plant	3.95 ± 0.16	3.83 ± 0.25	3.85 ± 0.23	3.91 ± 0.20	3.91 ± 0.18
Protein	4.40 ± 0.12	4.06 ± 0.10	3.94 ± 0.07	3.99 ± 0.11	3.95 ± 0.02
Wine	0.63 ± 0.03	0.68 ± 0.14	0.67 ± 0.06	0.63 ± 0.05	0.64 ± 0.05
Yacht	0.72 ± 0.24	2.92 ± 3.06	2.52 ± 2.78	0.61 ± 0.22	0.62 ± 0.22

Table 5.2: Test-NLL (mean and std. dev. across 20 train-test splits) on UCI datasets, using deep ensembles (DE), deep kernel learning (DKL) and deep ensemble kernel learning (DEKL). Suffices -I and -O indicate using KL_I (Eq. (5.9)) and KL_O (Eq. (5.8)), respectively.

Dataset	DE	DKL-I	DKL-O	DEKL-I	DEKL-O
Boston housing	2.51 ± 0.23	2.59 ± 0.19	2.66 ± 0.40	2.54 ± 0.21	2.54 ± 0.21
Concrete	3.08 ± 0.28	3.01 ± 0.14	2.99 ± 0.13	2.94 ± 0.11	2.95 ± 0.12
Energy	1.66 ± 1.38	1.13 ± 0.55	1.24 ± 0.66	0.68 ± 0.09	0.67 ± 0.07
Kin8nm	-1.26 ± 0.02	-1.13 ± 0.04	-1.12 ± 0.05	-1.16 ± 0.02	-1.16 ± 0.02
Naval Propulsion	-6.69 ± 0.13	-5.65 ± 4.24	-6.28 ± 1.01	-6.52 ± 0.96	-6.78 ± 0.60
Power plant	2.77 ± 0.05	2.76 ± 0.07	2.77 ± 0.06	2.78 ± 0.06	2.78 ± 0.05
Protein	2.75 ± 0.05	2.82 ± 0.03	2.79 ± 0.02	2.81 ± 0.03	2.79 ± 0.01
Wine	0.96 ± 0.10	1.03 ± 0.23	1.01 ± 0.09	0.96 ± 0.07	0.97 ± 0.07
Yacht	0.21 ± 0.17	2.06 ± 1.33	1.78 ± 0.81	1.07 ± 0.06	1.08 ± 0.07

matrix of its learners; we say that the ensemble is more diverse if the learners have lower correlation. We base this notion of diversity on the work of Brown et al. [22], who showed that extending the classic bias-variance decomposition of the mean squared error to an ensemble of learners requires an additional term in the decomposition that quantifies the average covariance between distinct learners.

For each UCI dataset and each train-test split, we load the DE and DEKL models (with and without optimal prior covariance) already trained as in Sec. 5.3.2 and evaluate their feature matrices on the entire dataset (including both training and test points). Each row of a feature matrix corresponds to an observation, and each column corresponds to the output of one of five learners. For DE, we use the mean output of each learner as one of the features. For DEKL, we multiply each column of the feature matrix ($\Phi(X;\theta)$ in the notation of Sec. 5.2.2 with X the dataset) by the corresponding element of the parameter mean μ of the final BLM layer, and we additionally scale all columns by the number of learners (five); doing this, the DE and DEKL feature matrices are comparable in the sense that the predictive means of both models can be obtained as the average of the feature matrix columns. Finally, for each dataset, train-test split, and model, we calculate the correlation matrix of the feature matrix and then average all correlation matrices across the train-test splits; the result is one (mean) correlation matrix for each dataset and model.

We visualize the correlation matrices in Fig. 5.3. We show only two datasets here for brevity; results for all datasets may be found in Appendix E. Overall, DEKL results in lower correlation between learners and thus greater diversity, as hypothesized, and using the optimal prior covariance often boosts diversity further. However, more diversity does not necessarily imply better predictive performance, as the results on the Kin8nm and Naval propulsion datasets indicate. Elucidating the relationship between diversity and predictive performance is a topic we leave for future work.

5.4 Discussion

We have introduced the DEKL model: a special case of DKL that is more easily parallelizable and thus more efficient than other DKL models of comparable network complexity. We have seen that DEKL often achieves better predictive performance than deep ensembles, which are also highly parallelizable. Even with a linear base kernel, we have shown that DEKL is a universal kernel approximator if the feature network is allowed to be arbitrarily wide, and we handle the complexity of the feature network by partitioning it into an ensemble of learners. In our experiments, we found that DEKL often outperforms DE and promotes more diversity, suggesting that jointly training learners in a Bayesian framework can be beneficial. Interestingly, we also observed that DEKL often outperforms DKL of the same width, suggesting that partitioning the feature network not only grants computational efficiency through model parallelism but may also boost performance by reducing the parameter count compared to a fully-connected DKL.

We are considering several avenues for future work. We plan to study the scalability of DEKL to a very large number of learners in a distributed setting, which may be necessary to approximate a kernel of very high or infinite rank. A key challenge in this endeavor is that the size of the variational covariance Σ grows quadraticly with the number of learners;

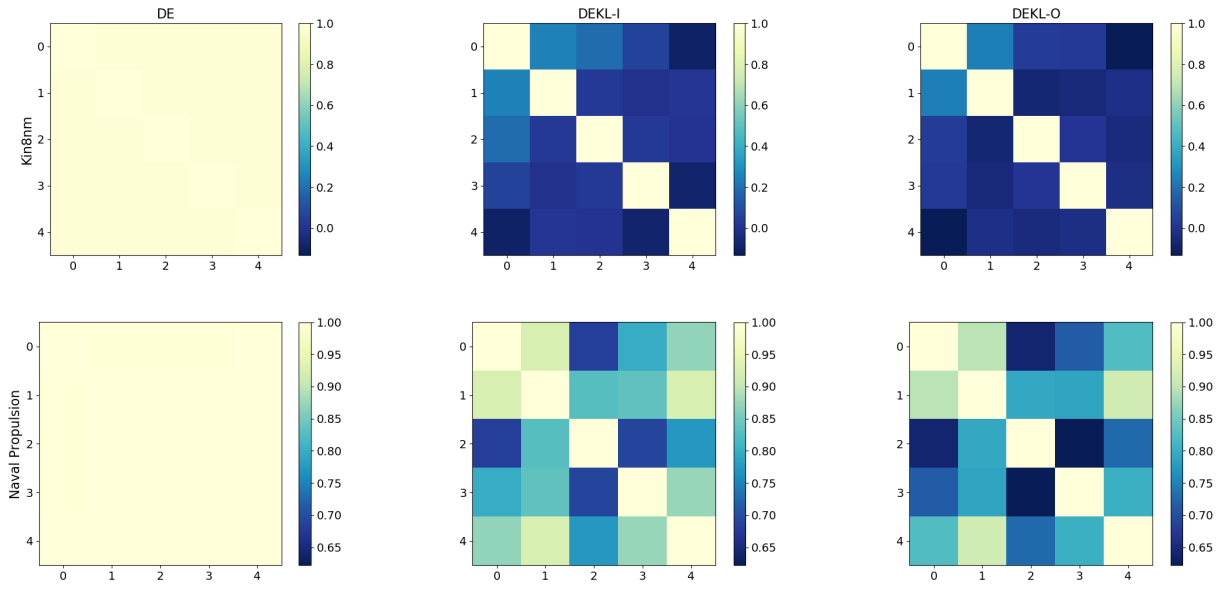


Figure 5.3: Correlation matrices of learners averaged over all train-test splits on the Kin8nm and Naval propulsion datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). Suffices -I and -O indicate using KL_I (Eq. (5.9)) and KL_O (Eq. (5.8)), respectively.

to deal with this, we will consider asymptotic regimes where efficient approximations of Σ are justified, namely the many-simple-learners regime and the complex-learners regime. In the first of these regimes, the dimension R of the span of H learners is much less than H, which allows us to assume that Σ is of low rank, specifically R. In the second regime, the learners may be sufficiently flexible to be orthogonal to one another, allowing us to assume Σ is diagonal. This could lead to an approach similar to that of Dasgupta et al. [38], but with distributed computation and using our variational framework. More generally, we may consider to approximate Σ as a sum of a low-rank matrix and a diagonal matrix, possibly striking a balance between the two asymptotic regimes.

Another interesting topic of investigation is the mechanism by which DEKL achieves performance often superior to that of DE. In DE, the learners are trained independently, and thus any diversity among the learners is solely due to random initialization and the nonconvexity of the loss function. In contrast, in DEKL, the learners are able to "communicate" with one another through the common final GP layer, and thus we hypothesize that the learners may "coordinate" with one another to ensure diversity. Indeed, in our experiments, we found that DEKL does lead to greater diversity, but the precise mechanism by which this diversity emerges and how it impacts predictive performance are less clear. We believe that diversity in DEKL may be linked to the posterior covariance, which is generally nonzero in this Bayesian setting.

It is also tempting to extend DEKL to deep Gaussian processes (DGPs), which are models defined as compositions of GPs [36, 33, 129]. This will result in a DGP inference method that again does not require an inducing points approximation, similar to the random features expansion approach of Cutajar et al. [33], except that the features are trainable DNNs that need not be orthogonal. Given that DEKL is a universal kernel approximator, we conjecture that a DGP with DEKL kernels can approximate any DGP with continuous kernels, under some distance metric on stochastic processes. Such a result would suggest a universal model for stochastic processes (or some suitably well-behaved subspace of them), arguably the most general universality theorem we can imagine in predictive modeling.

Chapter 6

Wide Neural Networks with Bottlenecks are Deep Gaussian Processes

Disclosure

A version of this chapter has been published in the Journal of Machine Learning Research [1].

Devanshu Agrawal and Jacob Hinkle conceptualized the paper. Devanshu Agrawal, Theodore Papamarkou, and Jacob Hinkle designed the experiments. Devanshu Agrawal proved the theorems and ran the experiments. Devanshu Agrawal wrote the manuscript with support from Theodore Papamarkou and Jacob Hinkle. All authors read and approved the manuscript.

The only revisions made to this chapter since the original publication are formatting and renaming the last section.

Abstract

There has recently been much work on the "wide limit" of neural networks, where Bayesian neural networks (BNNs) are shown to converge to a Gaussian process (GP) as all hidden layers are sent to infinite width. However, these results do not apply to architectures that require one or more of the hidden layers to remain narrow. In this paper, we consider the wide limit of BNNs where some hidden layers, called "bottlenecks", are held at finite width. The result is a composition of GPs that we term a "bottleneck neural network Gaussian process" (bottleneck NNGP). Although intuitive, the subtlety of the proof is in showing that the wide limit of a composition of networks is in fact the composition of the limiting GPs. We also analyze theoretically a single-bottleneck NNGP, finding that the bottleneck induces dependence between the outputs of a multi-output network that persists through extreme post-bottleneck depths, and prevents the kernel of the network from losing discriminative power at extreme post-bottleneck depths.

6.1 Introduction

Deep neural networks have found great empirical success, achieving state-of-the-art performance on a variety of tasks such as those in computer vision and natural language understanding [84, 9, 97]. There is considerable interest in understanding the theoretical aspects of deep neural networks both to establish guarantees on the behavior of these models on certain classes of problems as well as to guide architecture design and optimization. One avenue of pursuit in this endeavor leads to the study of Bayesian neural networks (BNNs), where the parameters of the network are random variables following some probability distributions. BNNs thus bring the formalism and machinery of probability theory to bear on neural networks.

It is a foundational result that a BNN converges to a Gaussian process (GP) in the "wide limit"—i.e., as the widths of all hidden layers are sent to infinity while the prior distributions on weights are sharpened accordingly [113]. The resulting GP is called a "neural network Gaussian process" (NNGP). Although NNGP limits have been derived from various BNN architectures, they cannot be obtained from architectures requiring some hidden layers to remain narrow, such as certain autoencoders. It seems intuitive that the wide limit of a BNN with some hidden layers restricted to finite-width "bottlenecks" is a composition of

NNGPs, but until now this claim has not been proven. Such a composition of GPs is called a "deep Gaussian process" (DGP) in the literature [36]. Although DGPs were inspired by the compositional structure of deep neural networks, their connection to BNNs has not been established formally.

In this paper, we give a formal proof of the convergence of BNNs with bottleneck layers to a DGP in the wide limit, where the DGP is a composition of NNGPs. In doing so, we unify the two major approaches to making GPs "deeper"—NNGPs and DGPs, thus allowing NNGPs to be examined in the DGP framework. We will refer to the limiting DGP as a "bottleneck NNGP". Even though the result is intuitive, the proof is nontrivial as it requires us to formally justify that the limit of a composition of BNNs equals the composition of the limiting NNGPs.

In the bottleneck NNGP limit, we consider a sequence of BNNs all having the same architecture except that some hidden layers are growing to infinite width (Fig. 6.1). We call the hidden layers held to finite widths "bottleneck layers" or simply "bottlenecks", and we call each network in the sequence a "bottleneck BNN". We use the term "component" to refer to any subnetwork that is either (1) between the input layer and the first bottleneck layer, (2) between two bottleneck layers with no bottlenecks in between, or (3) between the last bottleneck layer and the output layer; each BNN is thus a composition of components, and each component maintains constant input and output dimensions with only its hidden layers growing in width over the sequence of networks.

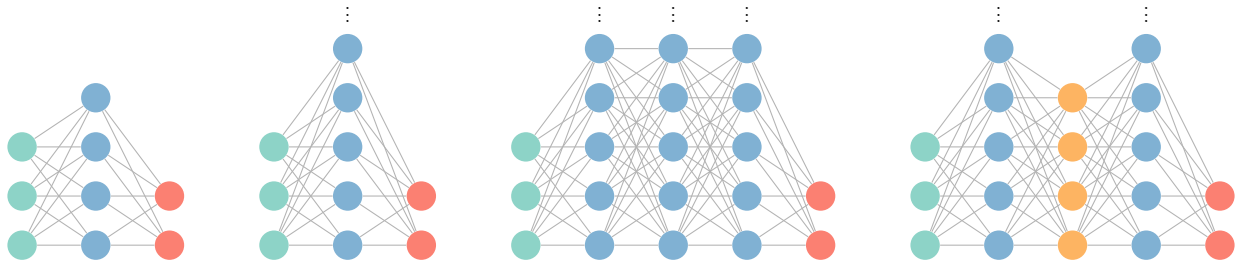
We know that each sequence of corresponding components converges to an NNGP in the wide limit. It is therefore intuitive to expect that the sequence of bottleneck BNNs (each BNN being a composition of components) converges to the composition of NNGPs—i.e., a bottleneck NNGP. However, this fact is not immediate, and care must be taken to verify that the limit procedure can be exchanged with the composition of components. In particular, we find that this exchangeability holds if each post-bottleneck component converges to an NNGP with a sufficient amount of uniformity with respect to its inputs.

We demonstrate the utility of bottleneck NNGPs and their link to no-bottleneck NNGPs empirically, showing that restricting a hidden layer of an NNGP to a bottleneck can boost its model likelihood on three example datasets[1].

We also characterize the effect of a bottleneck layer theoretically by analyzing an example multi-output single-bottleneck NNGP with rectified linear unit (ReLU) activation. We find that the bottleneck induces dependence between distinct response variables and derive a closed-form expression for the correlation between the squares (i.e., quadratic correlation) of the response variables. We show that in the deep post-bottleneck limit (infinitely many infinite-width hidden layers after the bottleneck), the quadratic correlation tends to 0 when the network is in the "disordered phase"—so that response variables decouple—but remains a nontrivial function of the inputs in the "ordered phase"—so that information about the inputs can be recovered. We identify the prior variance of the network weights as the order parameter responsible for the phase transition.

Similarly, in the deep post-bottleneck limit, we obtain a closed-form expression for the quadratic correlation of outputs of a single response variable given two inputs. We find that

[1]Code for our simulations and experiments is available at https://code.ornl.gov/d0a/bottleneck_nngp.



(a) MLP **(b)** NNGP **(c)** Deep NNGP **(d)** Bottleneck NNGP

Figure 6.1: Depiction of various NNGP architectures with three predictors (green nodes) and two response variables (red nodes). Blue nodes indicate hidden layers, with ellipses indicating layers that increase in width toward infinity. In our bottleneck NNGP model, one or more bottlenecks (finite-width hidden layers with orange nodes) are surrounded by infinite-width hidden layers. In the historical development of NNGP architectures, MLPs (sub-figure (a)) have been succeeded by shallow NNGPs (sub-figure (b), see Neal [113]), which in turn have been succeeded by deep NNGPs (sub-figure (c), see Lee et al., 2017, Matthews et al., 2018). In our paper, we propose bottleneck NNGPs (sub-figure (d)).

the quadratic correlation is 100% in the "disordered phase"—so that the network has lost all discriminative power at infinite depth—but is surprisingly a nontrivial function of the inputs in the "ordered phase". This behavior in the ordered phase stands in stark contrast to that of no-bottleneck NNGPs and indicates that bottleneck layers are essential for a very deep network to maintain discriminative power.

6.2 Preliminaries

In this section, we review prior work on DGPs and NNGPs to contextualize and motivate the bottleneck NNGP model. We also review the main theorem of Matthews et al., 2018 in Sec. 6.2.3, introducing notation that will be essential to stating our main result in Sec. 6.3.1.

6.2.1 Deep Gaussian processes

Compositions of GPs are known as deep Gaussian processes (DGPs) in the literature and were originally motivated by the success of deep neural networks and the hope to obtain similar success on small data sets where Bayesian methods generally shine [36]. DGPs have indeed been shown to outperform shallow GPs on a variety of regression and classification tasks [36, 129]. Damianou and Lawrence [36] implement DGPs in a sparse variational inducing points framework based on the work of Titsias [139] in order to simplify the composition of GPs to a set of separate but coupled GPs, but their implementation is restricted to small data sets with only a few hundred entries. Much of the DGP literature has therefore been dedicated to developing more efficient and scalable implementations [65, 35, 23, 146, 129]. Salimbeni and Deisenroth [129] in particular show that DGPs can be put into a stochastic variational framework as in Hensman et al. [64], allowing the models to be applied to much larger data sets.

DGPs offer the additional advantage that they can capture correlation between multiple outputs [146]. In contrast, the outputs of a shallow multi-output GP are by default independent, which can limit predictive performance for multi-output problems. Although methods have been proposed to model correlation in a shallow multi-output GP, such as through linear mixing of latent outputs [20, 7], DGPs capture correlation naturally through shared feature representations in the latent "bottleneck space"—similar to the approach taken with multi-task deep neural networks [127]. DGPs have therefore been applied to problems that can benefit from modeling correlation between multiple outputs, such as multi-task regression [4] and tasks involving partially observed multivariate outputs—i.e., missing values [146].

The mechanism by which the outputs of a DGP are made dependent predates the DGP model itself, as it was first introduced in the context of the Bayesian Gaussian process latent variable model (GP-LVM), where a Gaussian prior is placed on the latent inputs of a GP [88, 140]; for regression, each input is concatenated with such a latent random variable before it is fed into the GP [44]. The outputs then become dependent through their dependence on the common set of latent random variable inputs, which are analogous to the bottleneck activations of a DGP. However, it still remains quantitatively unclear how the introduction of a bottleneck layer—or in the case of an NNGP, the restriction of

a layer to finite width—induces correlation between multiple outputs under the prior and how this translates to correlation under the posterior. Moreover, although this mechanism is well-established for the GP-LVM and DGP models, it is conspicuously absent in the NNGP literature and thus its implications for NNGPs are not fully understood.

The DGP prior has been studied by Lu et al. [101], who show that for a single-bottleneck DGP with a single response variable, the prior has heavy tails, in contrast to shallow GPs. Their calculation of the prior kurtosis is similar to that of the quadratic correlation between distinct response variables of a multi-output DGP, but this connection is not discussed. Moreover, they only consider a bottleneck of width one and primarily focus on stationary kernels that do not arise from NNGP limits with common activation functions.

There is considerable interest in understanding the "deep limit" of DGPs—i.e. when arbitrarily many GPs are composed together. Duvenaud et al. [45] and Dunlop et al. [43] show that DGPs with a certain class of kernels have trivial, pathological, or convergent deep limits, meaning that increasing the depth of a DGP beyond some point is either detrimental to performance or diminishingly beneficial. However, they do not consider NNGP kernels and thus do not analyze deep limits of architectures with both bottlenecks and infinitely many infinite-width hidden layers.

Although DGPs were inspired by deep neural networks, there is little literature concretely establishing their connection. Duvenaud et al. [45] discuss the connection between DGPs and neural networks at a high level to motivate studying the deep limit of DGPs with radial basis function (RBF) kernels, but the implication for neural networks is not treated formally. Gal and Ghahramani [48] consider a DGP where the kernel of each GP layer is an integral as in Williams [151]. They show that a Monte Carlo estimation of the kernels leads to a BNN approximation of the DGP, where the width of a hidden layer corresponds to the size of the Monte Carlo sample. However, they do not formally verify convergence of a BNN to a DGP in the limit of infinite width. Moreover, their bottleneck layers have no activation function and are not scaled to allow an NNGP to be recovered as the bottlenecks are sent to infinite width.

6.2.2 Wide neural networks as GPs

A foundational result in the study of BNNs came when Neal [113] showed that a BNN with one hidden layer converges to a GP in the "wide limit"—i.e., as the number of hidden neurons is sent to infinity. Shortly after, Williams [151] derived analytic expressions for the kernel of the GPs corresponding to neural networks with sigmoidal and Gaussian hidden units. These works connected neural networks to the world of Bayesian nonparametrics and kernel methods and thus offered a new perspective to interrogate and probe the behavior of neural networks. In particular, while training neural networks is challenging since it requires the optimization of highly non-convex objective functions, GPs are nonparametric models that admit exact Bayesian inference, where the predictive posterior distribution can be written in closed form [125].

Since the works of Neal [113], Williams [151], new insights into BNNs have steadily emerged. Cho and Saul [28] interpreted a BNN as a feature embedding map and derived the equations for the propagation of a kernel through the layers of a deep neural network with rectified polynomial unit activations. Subsequent works built upon these findings to

elucidate key theoretical aspects of neural networks including expressivity [121], generalization power [62], initialization [37], and trainability [130]. More recently, the original result by Neal [113] has been extended to deep architectures by showing that a deep BNN converges to a GP as the widths of all hidden layers are simultaneously sent to infinity [Lee et al., 2017, Matthews et al., 2018]. We refer to GPs that arise from such a limit as "neural network Gaussian processes" (NNGPs). As the work of Matthews et al., 2018 illustrates, this extension is nontrivial; the proof by Neal [113] relies on the Central Limit Theorem, but the assumption of independent and identically distributed (IID) random variables necessary for the Central Limit Theorem does not hold for deep architectures. The proof by Matthews et al., 2018 for deep architectures is instead based on a more exotic central limit theorem as given in Blum et al. [18].

Since the extension of the NNGP limit to deep architectures, there have been a number of works establishing and analyzing analogous wide-limit results for more modern neural network architectures that are used in practice today. These include convolutional neural networks [54, 117], weight-tied autoencoders [95], and most generally any network that can be represented as a "tensor program"—including recurrent neural networks and attention networks among others [159]. Alongside these works, new insights into the trainability and generalization power of neural networks have continued to emerge, based on the tractable learning dynamics of neural networks in the wide limit [75, 92, 10].

One application of the NNGP limit that is of particular note is that it can make the analysis of the "deep limit"—i.e., as the number of hidden layers is sent to infinity—tractable [121, 130, 161, 89]. Poole et al. [121] and Schoenholz et al. [130] show that the correlation between two inputs transformed through an NNGP with sigmoidal activation function has a fixed point at 100% in the deep limit that transitions from stable to unstable (i.e., ordered to chaotic) when the variances of the Gaussian weights and biases cross a certain phase boundary; the network is shown to be highly expressive in the chaotic phase and optimally trainable near the phase boundary. In contrast, for networks with rectified linear unit (ReLU) activations, the correlation between transformed inputs has a stable fixed point at 100% regardless of the weight variance, implying that an NNGP with ReLU activation has no discriminative power at infinite depth [89].

The works described above all consider BNNs in the wide limit, and thus the results and insights therein do not apply to neural network architectures that require one or more finite width or "bottleneck" layers. One of the most important classes of neural networks that frequently require bottleneck layers is that of autoencoders [69, 80]. Another example is neural networks with a word embedding layer, which is currently key to the successful application of neural networks to natural language understanding [111]. Both word embedding layers and many autoencoder models aim to find dense feature representations and therefore depend on low-dimensional spaces. Even for neural network architectures that are not directly meant for dense representation learning, it has still been argued and demonstrated that bottleneck layers perform data compression and therefore help to boost generalization power [138]. Particularly for fully-connected architectures, which is what we consider in this work, it has been shown that the insertion of linear bottleneck layers between two linear ReLU layers boosts predictive performance by reducing sparsity and improving gradient flow [98]. This prompts the question: How can insights based on very wide BNNs be generalized to networks in which one or more hidden layers are held fixed to a finite width? The first step in addressing

this question is to understand what happens if we let all but finitely many hidden layers of a BNN grow to infinite width. We call these finite-width hidden layers "bottleneck layers". It is already established that the component networks between consecutive bottleneck layers converge to GPs, and thus we intuitively expect a BNN with bottleneck layers to converge in the wide limit to a composition of GPs. In Sec. 6.3.1, we formally verify that this is the case.

6.2.3 The no-bottleneck NNGP limit

The bottleneck NNGP limit is a generalization of the (no-bottleneck) NNGP result proved by Matthews et al., 2018. Moreover, one component of our proof is verifying that BNNs converge in distribution uniformly on compact sets, and our approach to proving this closely follows the proof of Matthews et al., 2018. In this section, we state the NNGP limit result by Matthews et al., 2018, which also allows us to introduce key concepts and notation along the way.

We consider a traditional fully-connected network mapping $\mathbb{R}^M$ to $\mathbb{R}^L$ with D hidden layers and nonlinearity ϕ. Let H_μ be the width of the μ-th hidden layer. The propagation of an input x through the network is then governed by a recursion with initial step

$$f_i^{(1)}(x) = b_i^{(1)} + \sum_{j=1}^{M} w_{ij}^{(1)} x_j, \tag{6.1}$$

and for $\mu \in \{1, \ldots, D\}$,

$$g_i^{(\mu)}(x) = \phi[f_i^{(\mu)}(x)], \tag{6.2}$$

$$f_i^{(\mu+1)}(x) = b_i^{(\mu+1)} + \sum_{j=1}^{H_\mu} w_{ij}^{(\mu+1)} g_j^{(\mu)}(x). \tag{6.3}$$

In Eq. (6.2), i ranges from 1 to H_μ. In Eq. (6.3), i ranges from 1 to $H_{\mu+1}$ for $\mu = 1, \ldots, D-1$, and from 1 to L for $\mu = D$. We refer to $f^{(\mu)}(x)$ and the $g^{(\mu)}(x)$ as the preactivations into and activations out of the μ-th hidden layer, respectively. The top-most preactivations $f^{(D+1)}(x)$ are the outputs of the network.

We require mild assumptions on the nonlinearity ϕ for our main theorem to hold; these are the same assumptions made by Matthews et al., 2018, namely the linear envelope condition.

Definition 6.1 (Linear envelope condition)**.** *A nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$ is said to satisfy the linear envelope condition if it is continuous and there exist positive constants C and M such that*

$$|\phi(x)| < C + M|x| \text{ for all } x \in \mathbb{R}.$$

Many popular activation functions such as tanh, ReLU, and leaky ReLU satisfy the linear envelope condition, and thus our result is quite general with regards to the choice of nonlinearity.

We now turn the above network into a random network by placing IID normal distributions on the weights $w^{(\mu)}$ and biases $b^{(\mu)}$ of the network:

$$w_{ij}^{(\mu)} \sim \mathcal{N}\left(0, \frac{v_w^{(\mu)}}{H_{\mu-1}}\right), \tag{6.4}$$

$$b_i^{(\mu)} \sim \mathcal{N}(0, v_b^{(\mu)}), \tag{6.5}$$

where we set $H_0 = 1$ for the purpose of defining these distributions. The variance of the weights after the first layer are scaled inversely to the preceding hidden layer width so that the Central Limit Theorem can be applied to the convergence of BNNs to a GP. With a slight abuse of terminology, we will call the constants $v_w^{(\mu)}$ and $v_b^{(\mu)}$ "weight and bias variance hyperparameters" even though $v_w^{(\mu)}$ is not the actual variance of the weights.

The output $f^{(D+1)}(x)$ is now a random vector of dimension L for each input x, and we therefore understand a BNN as an instance of a stochastic process. We give a formal definition next, after we introduce some notation. If Ω is a probability space and E is a measurable space, then an E-valued stochastic process F with index set X is a function $F : X \times \Omega \mapsto E$ such that $F(x, \cdot)$ is a measurable function for each $x \in X$. By the notation $F(x)$, we refer to the random variable $F(x) : \Omega \mapsto E$ defined by $F(x)(\omega) = F(x, \omega)$.

Definition 6.2 (Bayesian neural network)**.** *A Bayesian neural network (BNN) F mapping $\mathbb{R}^M$ to $\mathbb{R}^L$ with D hidden layers of widths H_μ, $\mu \in \{1, \ldots, D\}$, and nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$ is a stochastic process $F : \mathbb{R}^M \times \Omega \mapsto \mathbb{R}^L$ defined such that $F(x) = f^{(D+1)}(x)$, where $f^{(D+1)}$ is the neural network output defined through the recursion of preactivations and activations of Eqs. (6.2)-(6.3).*

Matthews et al., 2018 prove the following theorem concerning the convergence of BNNs with no bottleneck layers.

Theorem 6.3 (NNGP theorem, Matthews et al., 2018)**.** *Let $\{F[n]\}_{n=1}^{\infty}$ be a sequence of BNNs mapping $\mathbb{R}^M$ to $\mathbb{R}^L$ each with D hidden layers of widths $H_\mu[n]$, $\mu \in \{1, \ldots, D\}$, and nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$ that satisfies the linear envelope condition. If $H_\mu[n]$ is strictly increasing in n for each μ, then $\{F[n]\}_{n=1}^{\infty}$ restricted to a countable index set $\mathcal{X} \subset \mathbb{R}^M$ converges in distribution to the Gaussian process $\mathcal{GP}(0, K)$, where K is a kernel defined recursively by Eqs. (6.6)-(6.7).*

Note that we use the suffix $[n]$ instead of a subscript to index a sequence of stochastic processes. Convergence in distribution is defined in the measurable space $((\mathbb{R}^L)^\infty, \mathcal{A})$ of $\mathbb{R}^L$-valued sequences; details are provided in Appendix F. The limiting GP in Thm. 6.3 is called a neural network Gaussian process (NNGP). If $f_i^{(\mu)}$ is the limiting NNGP of $\{f_i^{(\mu)}[n]\}_{n=1}^{\infty}$, then the NNGP kernel is defined through a recursion with initial step

$$K_{ij}^{(1)}(x_1, x_2) = \mathrm{E}[f_i^{(1)}(x_1) f_j^{(1)}(x_2)] = \delta_{ij}(v_b^{(1)} + v_w^{(1)} x_1 \cdot x_2), \tag{6.6}$$

and for $\mu \in \{1, \ldots, D\}$,

$$\begin{aligned} K_{ij}^{(\mu+1)}(x_1, x_2) &= \mathrm{E}[f_i^{(\mu+1)}(x_1) f_j^{(\mu+1)}(x_2)] \\ &= \delta_{ij}\left(v_b^{(\mu+1)} + v_w^{(\mu+1)}\, \mathrm{E}_{z_1, z_2 \sim \mathcal{N}(0, C^{(\mu)})}[\phi(z_1)\phi(z_2)]\right), \end{aligned} \tag{6.7}$$

where $C^{(\mu)}$ is the 2×2 matrix with entries $c_{ab}^{(\mu)} = K_{11}^{(\mu)}(x_a, x_b)$; here we could have used $K_{ii}^{(\mu)}$ in place of $K_{11}^{(\mu)}$ for any i since the NNGP preactivations $f_i^{(\mu)}(x)$ into the μ-th hidden layer are IID over i. The (countably infinite) kernel matrix $K^{(\mu)}(\mathcal{X}, \mathcal{X})$ is therefore block-diagonal with the (i,j)-th block $K_{ij}^{(\mu)}(\mathcal{X}, \mathcal{X})$.

The kernel $K : \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}^{L \times L}$ in Thm. 6.3 is given by $K = K^{(D+1)}$. Observe that the L outputs of the BNNs converge to IID GPs so that all correlations between the outputs of the networks are lost in the infinite width limit. We will see that bottleneck layers help to preserve some correlations between outputs.

6.3 The bottleneck NNGP theorem

In this section, we state and prove the bottleneck NNGP theorem, we show that a single-bottleneck NNGP approximates a no-bottleneck NNGP as the bottleneck width is increased, and we explore the effect of depth and width on bottleneck NNGPs using three example datasets.

6.3.1 Statement of main theorem

We now state our main theorem, which is a direct generalization of Thm. 6.3 to compositions of BNNs. Given two stochastic processes $F^{(1)} : X \times \Omega^{(1)} \mapsto Y$ and $F^{(2)} : Y \times \Omega^{(2)} \mapsto Z$, we define the composition $F^{(2)} \circ F^{(1)}$ as the stochastic process $F^{(2)} \circ F^{(1)} : X \times (\Omega^{(1)} \times \Omega^{(2)}) \mapsto Z$ with

$$(F^{(2)} \circ F^{(1)})(x, (\omega_1, \omega_2)) = F^{(2)}(F^{(1)}(x, \omega_1), \omega_2).$$

Theorem 6.4 (Bottleneck NNGP theorem)**.** *Let $\{B_d \in \mathbb{N}\}_{d=0}^D$ for $D \in \mathbb{N}$ with $B_0 = M$ and $B_D = L$. For each $d \in \{1, \ldots, D\}$, let $\{F^{(d)}[n]\}_{n=1}^\infty$ be a sequence of BNNs mapping $\mathbb{R}^{B_{d-1}}$ to $\mathbb{R}^{B_d}$ with D_d hidden layers of widths $H_\mu^{(d)}[n]$, $\mu \in \{1, \ldots, D_d\}$, and nonlinearity ϕ that satisfies the linear envelope condition. If $H_\mu^{(d)}[n]$ is strictly increasing in n for each $d \in \{1, \ldots, D\}$ and $\mu \in \{1, \ldots, D_d\}$, then the sequence of bottleneck random neural networks $\{F^{(D)}[n] \circ \cdots \circ F^{(1)[n]}\}_{n=1}^\infty$ restricted to a countable index set $\mathcal{X} \in \mathbb{R}^M$ converges in distribution in $((\mathbb{R}^L)^\infty, \mathcal{A})$ to $F^{(D)} \circ \cdots \circ F^{(1)}$, where $F^{(d)}$ is the limiting NNGP of $\{F^{(d)}[n]\}_{n=1}^\infty$.*

Remark 6.5 (Nonlinear bottleneck)**.** *Theorem 6.4 as stated above assumes no nonlinearity on the bottleneck layers. However, the theorem also holds when we replace $F^{(d)}[n]$ and $F^{(d)}$ with $F^{(d)}[n] \circ \left(\frac{1}{\sqrt{B_{d-1}}}\phi\right)$ and $F^{(d)} \circ \left(\frac{1}{\sqrt{B_{d-1}}}\phi\right)$ respectively for $d \in \{2, \ldots, D\}$, that is when we scale the weights after each bottleneck layer by layer width in the same way as all other weights after a hidden layer and and we place nonlinearities ϕ on the bottleneck layers. The proof is nearly identical to the proof of Thm. 6.4 (see Remark .11 in Appendix G.2 for details).*

Remark 6.6 (Discontinuous nonlinearity)**.** *Theorem 6.4 holds even if the nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$ is continuous only almost everywhere, as long as ϕ is continuous at 0 or $v_b > 0$. Each of these two conditions ensures that the Continuous Mapping Theorem is still applicable in Lemmas .14-.16 (see Remark .12 in Appendix G.2 for details). This extends the class of allowable nonlinearities to include such prominent examples as the Heaviside step function used in the first perceptron model [126].*

Remark 6.7 (Converse of the main theorem)**.** *The converse of Thm. 6.4—that every DGP is the bottleneck NNGP limit of a BNN with IID priors and a nonlinearity satisfying the linear envelope condition—does not hold. A simple counterexample is a no-bottleneck single-hidden-layer NNGP with the rectified polynomial unit activation* $\phi(x) = \max(0, x)^n$ *for* $n \geq 2$ *[28]; this is a GP that can only result from a wide limit if the linear envelope condition is violated. A more trivial counterexample is any GP with a linear kernel* $k(x_1, x_2) = x_1^\top G x_2$ *where* G *is a symmetric positive-semidefinite matrix not proportional to the identity matrix. By Eq. (6.6), the NNGP kernel depends on its inputs only through their dot product and is thus invariant under rotations. The kernel* k *with metric* G *can therefore only arise from a wide limit if the "IID priors" condition is violated.*

Here we consider a sequence of BNNs with $D-1$ bottleneck layers of widths $B_1, \ldots, B_{D-1}$. As all hidden layers except the bottleneck layers tend to infinite width, each component network converges to an NNGP by Thm. 6.3, but it is less obvious that the composition of components tends to the composition of the limiting NNGPs. Our proof depends on several original lemmas (Lemmas .7-.10 in Appendix G.2). However, Lemma .10 is a simple generalization of Lemma 12 in Matthews et al., 2018, and its proof therefore runs in parallel to that in Matthews et al., 2018. The complete proof of the main theorem as well as proofs for all supporting lemmas can be found in Appendix G; we recommend readers to start at the introduction of Appendix G, where we provide a detailed sketch of the proof and discuss the high-level function of each lemma. Definitions and properties of the various modes of convergence of stochastic processes pertinent to the proof are discussed in Appendix F.

6.3.2 Correspondence to the no-bottleneck NNGP

We expect that in the limit as bottlenecks are sent to infinite width, the bottleneck NNGP converges to the (no-bottleneck) NNGP with the same number of hidden layers. The next theorem gives this result for the case of a bottleneck NNGP with one bottleneck layer.

Theorem 6.8 (Wide bottleneck correspondence)**.** *Let* $\{F^{(H)}\}_{H=1}^\infty$ *be a sequence of single-bottleneck NNGPs mapping* $\mathbb{R}^M$ *to* $\mathbb{R}^L$ *with* D_1 *hidden layers and* D_2 *hidden layers before and after the bottleneck of width* H *and with nonlinearity* $\phi : \mathbb{R} \mapsto \mathbb{R}$ *that satisfies the linear envelope condition. Suppose the nonlinearity* ϕ *is also applied to the bottleneck and that its activations are scaled by* $\frac{1}{\sqrt{H}}$*, in accordance with Remark 6.5. Suppose also that IID Gaussian noise* $\mathcal{N}(0, v_n)$ *is added to the networks for any* $v_n > 0$*. Then*

(a) $\{F^{(H)}\}_{H=1}^\infty$ *restricted to a countable index set* $\mathcal{X} \subset \mathbb{R}^M$ *converges in distribution in* $((\mathbb{R}^L)^\infty, \mathcal{A})$ *to an NNGP* F *with* $D_1 + D_2 + 1$ *hidden layers and Gaussian noise* $\mathcal{N}(0, v_n)$*.*

(b) For every finite set of inputs $X \subset \mathbb{R}^M$*, the sequence of probability density functions (PDFs) of* $\{F^{(H)}\}_{H=1}^\infty$ *converges pointwise to the PDF of the NNGP* F*.*

Remark 6.9. *Statement (a) of Thm. 6.8 holds even if there is no additive Gaussian noise* ($v_n = 0$)*; the proof uses the technique in the proof of Lemma .7, where the function* $X \to \Pr(F(X) \in U)$ *is shown to be continuous for an NNGP* F *and a continuity set* U*.*

The proof of Thm. 6.8 is given in Appendix H. It is based on the observation that the activations in the bottleneck layer of a single-bottleneck NNGP are IID. Since the post-bottleneck NNGP depends on these activations through their Gram matrix only and since the activations are inversely scaled by the bottleneck width, then the post-bottleneck NNGP is a function of the sample covariance of bottleneck activations, which converges to the pre-bottleneck NNGP kernel in the limit of infinite bottleneck width by the Law of Large Numbers. Extending this result to the case of multiple bottlenecks is left for future work.

6.3.3 Experiments

Statement (b) of Thm. 6.8 implies that the marginal log-likelihood (MLL) of a single-bottleneck NNGP architecture given data (X, Y) and fixed variance hyperparameters v_n, v_b, v_w converges to the MLL of the corresponding NNGP as the bottleneck is sent to infinite width. The MLL of the bottleneck NNGP is just the logarithm of the PDF given in Eq. (23) evaluated at the dataset. This formally validates the intuition that a bottleneck NNGP with a sufficiently wide bottleneck is a similar model to a no-bottleneck NNGP. However, the utility of bottleneck NNGPs with narrower bottlenecks as measured by MLL is less clear. We investigate this question on a simulated dataset that we call Rings and on two publicly available datasets—Fisher's Iris data set [8, 46] and the US Census Boston housing prices dataset [60].

The Rings dataset consists of 120 points lying on two interlocked cylindrical bands or "rings" orthogonal to one another and passing through one another's centers (Fig. 6.2). We took a regular 12×5 lattice of 60 points in $[0, 2\pi) \times \left[-\frac{1}{2}, \frac{1}{2}\right]$ and mapped it onto one of the rings in $\mathbb{R}^3$ by

$$(\theta, z) \mapsto (\cos\theta, \sin\theta, z).$$

To generate the second ring, we rotated a copy of the first by 90° in the xz-plane and translated it by 1 along the y-axis. We assigned a label of 0 to the 60 points on the first ring and a label of 1 to the 60 points on the second. The Rings dataset is therefore a non-linearly separable binary classification problem where the dimension of the data manifold is less than that of the linear span of the data points.

We also considered the Iris and Boston House-Prices datasets. Like Rings, Iris is a classification problem, but we one-hot encoded its labels to include a multivariate dataset with strongly correlated labels. For simplicity, following Lee et al. [89], we implemented all three problems as regression tasks. We standardized both the input and target sets of all three datasets to help place the three problems on similar scales, as well as to set a reasonable scale for the variance hyperparameters v_b and v_w.

We used a single-bottleneck NNGP with one infinitely wide hidden layer before the bottleneck, bottleneck width H, and post-bottleneck depth D (i.e., D infinitely wide hidden layers after the bottleneck) for various values of H and D. We equipped all hidden neurons with the normalized ReLU activation

$$\phi(x) = \sqrt{2}\max(0, x). \tag{6.8}$$

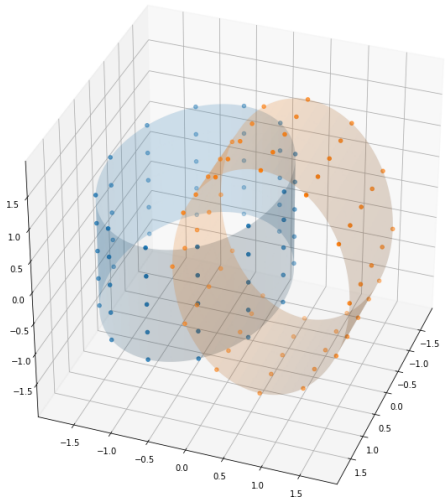


Figure 6.2: Visualization of the simulated Rings dataset. We take 60 regularly spaced points from each ring. We assign a binary label to each point based on the ring to which it belongs.

The propagation of the NNGP kernel through the hidden layers given in Eq. (6.7) admits a closed form for the normalized ReLU activation and is given by [28]:

$$K_{ij}^{(\mu+1)}(x_1, x_2) = \delta_{ij}\left[v_b + v_w \cdot \frac{1}{\pi}\sqrt{K_{11}^{(\mu)}(x_1, x_1)K_{11}^{(\mu)}(x_2, x_2)}J_1(\theta^{(\mu)})\right], \tag{6.9}$$

$$\theta^{(\mu)} = \cos^{-1}\left[\frac{K_{11}^{(\mu)}(x_1, x_2)}{\sqrt{K_{11}^{(\mu)}(x_1, x_1)K_{11}^{(\mu)}(x_2, x_2)}}\right],$$

where the function J_1 is defined as

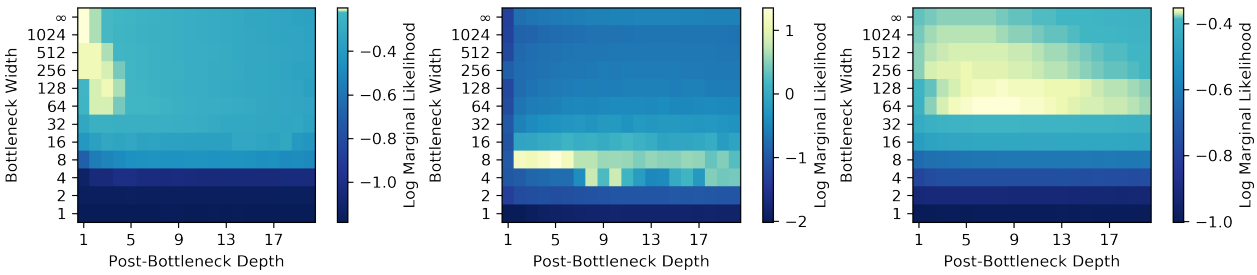
$$J_1(\theta) = \sin\theta + (\pi - \theta)\cos\theta.$$

Our goal is to find the bottleneck NNGP architecture (i.e., combination of bottleneck width H and post-bottleneck depth D) with the greatest likelihood given a dataset (X, Y) of N observations. We calculated the MLL of each bottleneck NNGP architecture (H, D) using

$$\text{MLL}(H, D; X, Y) = \log p(Y; X, H, D, v_{b*}(H, D), v_{w*}(H, D), v_{n*}(H, D)), \tag{6.10}$$

where on the right-hand side, p is the PDF in Eq. (23) of the data outputs given the data inputs and network architecture, and where the variance hyperparameters are set to their maximum likelihood estimates. We found the optimal variance hyperparameters iteratively through gradient descent. During the forward pass through the network in each iteration, we estimated the integral in Eq. (23) by drawing 100 IID Monte Carlo (MC) samples—each an $N \times H$ matrix with IID columns—from the pre-bottleneck NNGP. We did so using the local reparameterization trick [82], so that each sample is a transformation of a draw from the $(N \times H)$-dimensional standard normal distribution. We used the Adam optimizer [81] to take advantage of the gradient noise generated by MC sampling during optimization; we set the initial learning rate to 0.1. In order to ensure that the noise observed in the learning curves was due only to MC sampling and not due to a large learning rate, we decayed the learning rate as follows: After the backward pass of each iteration, we re-evaluated the MLL using the same draw from the $(N \times H)$-dimensional standard normal distribution for the MC samples; if the new MLL was less than the value obtained from the initial forward pass of the iteration, then we multiplied the learning rate by 0.9. We iterated the optimization procedure until convergence of the MLL learning curves; once complete, we evaluated Eq. (6.10) once more—this time with 1000 MC samples—to obtain the final MLL estimate for each network architecture.

On all three datasets, the maximum MLL is attained at a finite bottleneck width and post-bottleneck depth ($H_* = 1024$ and $D_* = 1$ for Rings; $H_* = 8$ and $D_* = 5$ for Iris; $H_* = 64$ and $D_* = 7$ for Boston), thus demonstrating the utility of bottleneck layers in NNGP models (Fig. 6.3). On Rings and Boston, we also observe that the optimal post-bottleneck depth conditional on a bottleneck width roughly decreases as the bottleneck width increases. Since the no-bottleneck NNGP kernel with ReLU activation is known to degenerate to a constant kernel with no discriminative power [89], it makes sense that a deeper network may require a narrower bottleneck to help information propagate through the network. Although not conclusive from the figures alone, this observation at least warrants further investigation. In Sec. 6.4, we do just that and find that when the variance hyperparameters are fixed, then the bottleneck width and post-bottleneck depth are indeed intimately related.



(a) Rings **(b)** Iris **(c)** Boston

Figure 6.3: Marginal log-likelihoods (MLL) of three datasets normalized by number of observations (data points) under a bottleneck NNGP for various bottleneck widths and post-bottleneck depths. Infinite bottleneck width corresponds to the limiting no-bottleneck NNGP. On all three datasets, the maximum MLL is attained at some finite bottleneck width and post-bottleneck depth.

6.4 Bottleneck layers induce dependence

In Sec. 6.3.3, we showed empirically that the model likelihood of an NNGP is improved if one of the hidden layers is restricted to a finite-width bottleneck and speculated that the optimal post-bottleneck depth may increase as the bottleneck is narrowed. In this section, we investigate possible mechanisms underlying these observed trends in model performance. We start by showing that although linear correlations between distinct response variables or "output neurons" of an NNGP remain zero even in the presence of bottlenecks, the corresponding *quadratic* correlations are often non-zero. We also analyze the behavior of this quadratic correlation in the deep post-bottleneck limit—i.e., as the post-bottleneck depth is sent to infinity. This deep limit is distinct from the ones typically considered for DGPs, where the number of GP components is sent to infinity, and for NNGPs, where the depth of a single GP component is sent to infinity. Proposition 6.14 provides a striking result, which implies that bottleneck layers help a network retain discriminative power even at extreme post-bottleneck depths.

Note that in this section, we primarily consider the ReLU activation defined in Eq. (6.8) as it is by far the most common nonlinearity used in deep learning today. In Sec. 6.4.5, we briefly consider other nonlinearities and contrast their deep limit behaviors with that of the ReLU activation, thereby highlighting the peculiarities of the ReLU activation.

6.4.1 An exact formula for quadratic correlation

The outputs of a multi-output GP prior are IID, and it follows that the outputs of the corresponding posterior remain independent (though not necessarily identically distributed). This is a limitation of GPs for multi-task learning applications, since information cannot be shared across tasks. One method to solve this problem was proposed by Bonilla et al. [20] who introduce a coupling matrix hyperparameter through which distinct output neurons or tasks can interact. This method, if applied to finite-width neural networks, would be superfluous since tasks could share information through a common set of features learned in the earlier layers of the network.

The key to correlating tasks in neural networks is clearly not depth alone, since the outputs of a (no-bottleneck) NNGP—however deep—are independent. Rather, following from the DGP framework as discussed in Sec. 6.2.1, the outputs of an NNGP become correlated if bottleneck layers are introduced, so that bottleneck NNGPs support multi-task learning out of the box. Correlation arising from finite-width bottleneck layers is exactly the type of behavior we expect in neural network architectures such as word embedding layers and many kinds of autoencoders, where the bottleneck width forces dense feature representations (i.e., feature representations that capture correlation) to be learned. The correlation structure that is induced in an NNGP prior through bottleneck layers is, however, subtle; distinct outputs of a bottleneck NNGP prior remain linearly uncorrelated (i.e., have zero covariance) but can be quadratically correlated (i.e., the squares of the outputs have non-zero covariance).

The expression for the quadratic correlation of outputs in a single-bottleneck NNGP prior can be obtained in closed form. Consider a bottleneck NNGP $F : \mathbb{R}^M \times \Omega \mapsto \mathbb{R}^2$ with one bottleneck layer of width H, any number of infinitely wide hidden layers before the bottleneck, and $D - 1$ infinitely wide hidden layers after the bottleneck. Suppose all hidden

neurons (including in the bottleneck) are equipped with the normalized ReLU activation defined in Eq. (6.8). We also scale the bottleneck activations by $\frac{1}{H}$ in accordance with Remark 6.5. Let v_b and v_w be the bias variance and weight variance hyperparameters of both the pre-bottleneck and post-bottleneck components of the bottleneck NNGP defined as in Eqs. (6.4)-(6.5). Suppose the bottleneck NNGP is fed two inputs $x_1, x_2 \in \mathbb{R}^M$. Then the preactivations into the H neurons in the bottleneck layer are IID with common 2D normal distribution $\mathcal{N}(0, C)$ for some covariance matrix

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}.$$

As is commonly done with DGPs [36, 129], we assume that IID Gaussian noise $\mathcal{N}(0, v_n)$ (with v_n arbitrarily small) is added to the preactivations of the bottleneck layer; We include the variance of the noise in the covariance matrix C, so that C is invertible. We also add IID Gaussian noise $\mathcal{N}(0, v_n)$ to each of the two outputs of the bottleneck NNGP.

It is easy to verify that the two outputs $F_1(x_a)$ and $F_2(x_b)$ (for any $a, b \in \{1, 2\}$) are linearly uncorrelated; this immediately follows from the conditional independence of outputs given the bottleneck activations. The relationship between the squares of the outputs is, however, less trivial. Let $Q^\times$ denote the matrix of correlations between the squares of the two outputs of F (the superscript $\times$ is used to emphasize that the correlation is between distinct output neurons).

Proposition 6.10 (Quadratic correlation between outputs)**.** *Consider the single-bottleneck NNGP F with normalized ReLU activation defined above. Then*

$$\begin{aligned} \operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2] &= \frac{v_w^{2D}}{H} \operatorname{Cov}_{(z,z')\sim\mathcal{N}(0,K)}[\phi(z)^2, \phi(z')^2] \\ &= \frac{v_w^{2D} c_{aa} c_{bb}}{H} \left(\frac{2}{\pi} J_2(\beta) - 1 \right), \end{aligned} \tag{6.11}$$

where $\beta = \frac{c_{ab}}{\sqrt{c_{aa} c_{bb}}}$ and the function J_2 is defined as [28]:

$$J_2(\beta) = 3 \sin\beta \cos\beta + (\pi - \beta)(1 + 2\cos^2\beta).$$

The corresponding correlation is

$$\begin{aligned} q_{ab}^\times &= \frac{\operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2]}{\sqrt{\operatorname{V}[F_1(x_a)^2] \operatorname{V}[F_2(x_b)^2]}} \\ &= \frac{\left(\frac{2}{\pi} J_2(\beta) - 1 \right)}{\sqrt{\left[15 + 2H \left(\frac{r_D}{c_{aa}} + 1 \right)^2 \right] \left[15 + 2H \left(\frac{r_D}{c_{bb}} + 1 \right)^2 \right]}}, \end{aligned} \tag{6.12}$$

where

$$r_D = \begin{cases} v_n + D v_b & \text{if } v_w = 1 \\ \frac{v_n}{v_w^D} + \frac{v_b}{1 - v_w} \left(\frac{1}{v_w^D} - 1 \right) & \text{otherwise.} \end{cases} \tag{6.13}$$

Remark 6.11 (Quadratic correlation for stationary kernels)**.** *In Prop. 6.10, if we instead consider a single-bottleneck DGP F where the post-bottleneck GP has a stationary kernel (such as the RBF kernel), then the quadratic correlation between outputs is $q^{\times}_{ab} = 0$. The non-stationarity of the NNGP kernel is therefore key to capturing some amount of correlation.*

The proof of Prop. 6.10 as well as the proofs of all other propositions in Sec. 6.4 are given in Appendix I. The significance of Prop. 6.10 is two-fold. First, quadratic correlation under the prior—although subtle—may translate to stronger dependence (such as linear correlation) under the posterior. Indeed, as stated in Remark 6.11, a DGP with RBF kernel captures less correlation under its prior than does a bottleneck NNGP with ReLU activation, and yet the former has been shown to be useful in modeling dependence in practice [4, 146]. This suggests that a bottleneck NNGP with ReLU activation may be just as useful in modeling dependence. Second, the ability of the network to capture quadratic correlation is closely linked to its ability to operate effectively at extreme depths; we discuss this in more detail in Sec. 6.4.4;

The pre-bottleneck component of the bottleneck NNGP is a map that sends input vectors in $\mathbb{R}^M$ to normally distributed real-valued random variables (preactivations of bottleneck neurons). By understanding covariance as an inner product on the space of finite-variance random variables, we can see that the covariance matrix C is the Gram matrix of bottleneck preactivations, and the angle β appearing in Prop. 6.10 is the angle between the preactivations of two inputs at one bottleneck neuron; we will call β the bottleneck angle.

The quadratic correlation (Eq. (6.12)) varies with the bottleneck width H roughly as $\frac{1}{H}$ and thus vanishes in the limit of infinite bottleneck width, recovering the independence of outputs of an NNGP with no bottlenecks. We empirically verified Eq. (6.12) for a bottleneck NNGP prior with one hidden layer before the bottleneck, one hidden layer after the bottleneck ($D = 2$), and with variance parameters $v_b = v_w = 1$ and $v_n = 10^{-4}$. We fed the example bottleneck NNGP two inputs $x_1 = (1, 0)$ and $x_2 = (0, 1)$. Then for each bottleneck width $H \in \{1, \ldots, 10\}$, we generated 10^6 IID samples of $(F_1(x_1), F_2(x_2))$ and used them to estimate $q^{\times}_{12}$. We repeated this simulation 10 times, and we report the mean estimate of $q^{\times}_{12}$ along with its standard deviation for each bottleneck width (Fig. 6.4). The empirical quadratic correlation estimates are very close to the theoretical values predicted by Eq. (6.12), with standard deviations all on the order of 10^{-3}.

We performed additional simulations to understand how multiple bottleneck layers affect the correlation of outputs, as we found this to be intractable theoretically. We still consider a bottleneck NNGP with $v_b = v_w = 1$ and $v_n = 10^{-4}$ that is fed two 2D inputs $x_1 = (1, 0)$ and $x_2 = (0, 1)$, but now we suppose the bottleneck NNGP has 11 hidden layers (including all bottleneck layers). We chose 11 hidden layers since it allows us to restrict zero to three hidden layers to bottlenecks such that the bottlenecks are equally spaced in depth. For each of the zero to three bottleneck layers, we ran the experiment described above for a single bottleneck and estimated the quadratic correlation $q^{\times}_{12}$ along with its standard deviation over ten runs for various bottleneck widths (all bottleneck layers have the same width). The quadratic correlation tends to zero with increasing bottleneck width regardless of the number of bottleneck layers, as we expect (Fig. 6.5). We additionally observe that for bottleneck widths $H \geq 2$, the quadratic correlation increases with the number of bottlenecks even if the

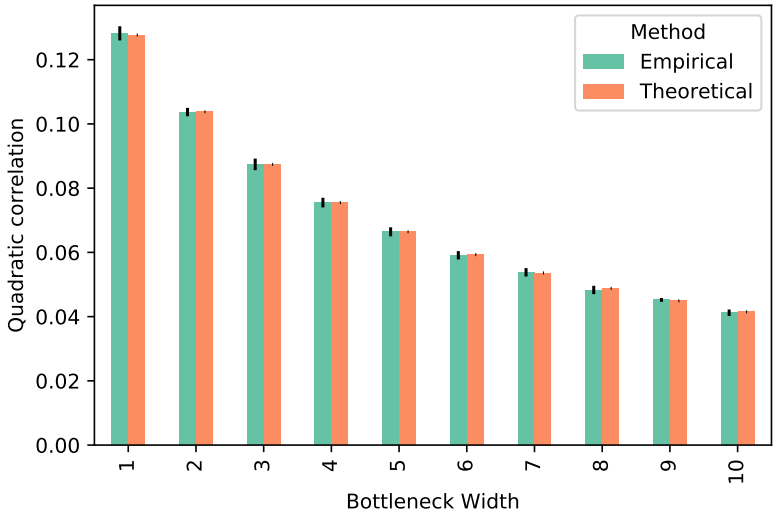


Figure 6.4: Empirical estimates with standard errors of the quadratic correlation of the outputs of an example bottleneck NNGP with one bottleneck surrounded by two infinite hidden layers, compared to theory (Eq. (6.12)) for various widths of the bottleneck layer. The theoretical values are all within one standard error of the empirical mean value.

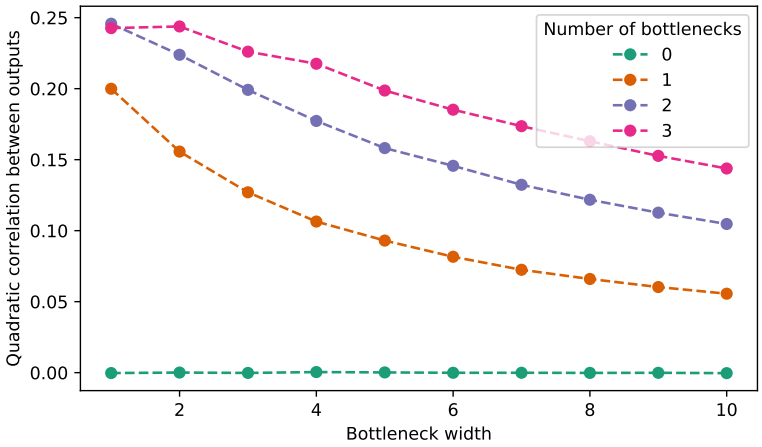


Figure 6.5: Empirical quadratic correlation of the outputs of an example bottleneck NNGP with 11 hidden layers, some of which are restricted to regularly spaced bottlenecks. Quadratic correlation increases with more numerous and narrower bottlenecks.

overall depth of the bottleneck NNGP remains the same, suggesting that bottlenecking more layers has a similar effect to further narrowing existing bottlenecks.

6.4.2 Quadratic correlation as a function of depth

The inverse dependence of the quadratic correlation on bottleneck width is intuitive since we know that the outputs of an NNGP are independent in the absence of bottlenecks. There is also an interesting and less obvious dependence of the quadratic correlation on the post-bottleneck depth D (where there are $D-1$ post-bottleneck hidden layers of infinite width) as well as on the angle β between the random bottleneck preactivations of the inputs x_a $(a=1,2)$ in the bottleneck layer, as captured by the covariance matrix C. We denote the quadratic correlation by $q_{ab}^{\times(D)}$ to make explicit its dependence on the post-bottleneck depth D and will sometimes write $q_{ab}^{\times(D)}(\beta)$ to further clarify its dependence on the bottleneck angle β. By Prop. 6.10, it is easy to verify that $q_{ab}^{\times(D)}(\beta)$ is strictly decreasing in β on $[0,\pi]$ with $\beta<\frac{\pi}{2}$ giving positive correlation, $\beta=\frac{\pi}{2}$ giving zero correlation, and $\beta>\frac{\pi}{2}$ giving negative correlation at all depths D. The quadratic correlation between outputs therefore encodes the correlation of inputs in the bottleneck layer. By Eq. (6.13), r_D is strictly increasing in D regardless of the values of $v_b, v_w > 0$. It follows that the absolute quadratic correlation $|q_{ab}^{\times(D)}(\beta)|$ strictly decreases with D for $\beta \neq 0$ (and remains 0 otherwise). A final property of the quadratic correlation is its range of possible values, which easily follows from the ranges of J_2 and r_D:

$$-\frac{1}{17} < q_{ab}^{\times(D)}(\beta) < \frac{5}{17}. \tag{6.14}$$

These properties are apparent in the plot of $q_{ab}^{\times(D)}(\beta)$ over D for an example single-bottleneck NNGP ($H=2$, $v_b=0.09$, $v_w=1.1$) with no pre-bottleneck hidden layers that is fed two inputs $x_1=(1,0)$ and $x_2=(\cos\alpha,\sin\alpha)$ for various values of the input angle α (Fig. 6.6); Since there are no pre-bottleneck hidden layers, the input angle α and bottleneck angle β are related through the equation

$$\cos\beta = \frac{v_b + v_w\cos\alpha}{v_b + v_w}.$$

Orthogonal bottleneck preactivations ($\beta=\frac{\pi}{2}$) and thus 0 quadratic correlation in the example bottleneck NNGP are then achieved at an input angle $\alpha \approx 0.526\pi$. Observe in general that if $v_b > 0$, then $\beta < \alpha$ and the range of β is strictly smaller than $[0,\pi]$, indicating that bias units promote positive quadratic correlation.

The behavior of r_D (Eq. (6.13)) in the limit of infinite post-bottleneck depth ($D\to\infty$) is easily analyzed. We have

$$r_\infty = \lim_{D\to\infty} r_D = \begin{cases} \frac{v_b}{v_w-1} & \text{if } v_w > 1 \\ \infty & \text{otherwise.} \end{cases} \tag{6.15}$$

This lets us determine what happens to the quadratic correlation of outputs as the number of post-bottleneck hidden layers grows to infinity.

Proposition 6.12 (Infinite-depth quadratic correlation between outputs)**.** *Consider the single-bottleneck NNGP F with normalized ReLU activation from Prop. 6.10, and suppose we send the post-bottleneck depth to infinity.*

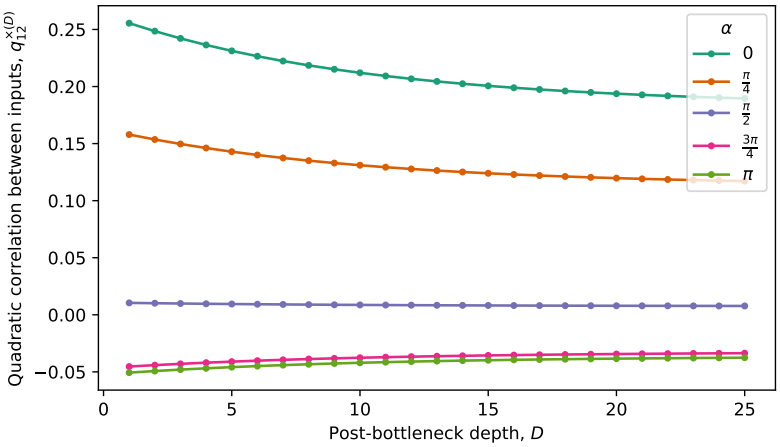


Figure 6.6: Theoretical quadratic correlation between outputs of a single-bottleneck NNGP over the post-bottleneck depth D for various angles α between the inputs. These quadratic correlations are asymptotically non-zero whenever $v_w > 1$ and the bottleneck preactivations (as random variables) of the inputs are not orthogonal (bottleneck orthogonality occurs at $\alpha \approx 0.526\pi$).

(a) The infinite-depth quadratic correlation matrix $Q^{\times(\infty)}$ has (a,b)-th element $q_{ab}^{\times(\infty)} = \lim_{D\to\infty} q_{ab}^{\times(D)}$ given by

$$q_{ab}^{\times(\infty)} = \begin{cases} \dfrac{\left(\frac{2}{\pi}J_2(\beta)-1\right)}{\prod\limits_{c=c_{aa},c_{bb}} \sqrt{15+2H\left(\dfrac{v_b}{(v_w-1)c}+1\right)^2}} & \text{if } v_w > 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6.16}$$

(b) Let G be the Gram matrix of inputs with entries $g_{ab} = x_a^\top x_b$. If $v_w > 1$, then the mapping $G \mapsto Q^{\times(\infty)}$ is invertible.

We visualize the infinite-depth quadratic correlation by plotting Eq. (6.16) as a function of v_w for a single-bottleneck NNGP ($H = 2$, $v_b = 0.09$) with no pre-bottleneck hidden layers that is fed two inputs $x_1 = (1, 0)$ and $x_2 = (\cos\alpha, \sin\alpha)$ for various values of the input angle α (Fig. 6.7). Note that the covariance matrix C and hence the bottleneck angle β are themselves functions of v_w, which is why we plot the quadratic correlation for fixed values of α instead of β. The infinite-depth quadratic correlation exhibits interesting behavior around $v_w = 1$; it is continuous but not differentiable at $v_w = 1$. The bottleneck NNGP therefore undergoes a phase transition at $v_w = 1$. The quadratic correlation tends to 0 in the $v_w \leq 1$ regime—meaning that the outputs of a bottleneck NNGP decouple (up through second order correlations) at infinite depth—while quadratic correlation is maintained through infinitely many, infinitely wide hidden layers in the regime $v_w > 1$, though even then the limiting correlation is weak (Eq. (6.14)).

A phase transition at $v_w = 1$ has already been noted in the literature in the behavior of no-bottleneck NNGP models at infinite depth [130, 121, 89]. Specifically, the kernel of an NNGP with normalized ReLU activation degenerates to a constant kernel at infinite depth with a value of either $\frac{v_b}{1-v_w}$ if $v_w < 1$ and ∞ otherwise. Bottleneck layers help to reveal a richer structure of this phase transition, as we explain next. Drawing an analogy to the classical Ising model in statistical mechanics [13], the hyperparameter v_w operates as an inverse temperature with a critical value at $v_w = 1$. The quadratic correlation is then an order parameter analogous to magnetization whose derivative contains a discontinuity at the phase boundary $v_w = 1$. In the $v_w < 1$ phase, the infinite-depth quadratic correlation is 0 regardless of the bottleneck angle β; information about the inputs into the bottleneck NNGP is therefore lost, analogous to a disordered system at large scale. However, as v_w crosses the phase boundary from below, the system undergoes a symmetry breaking with the infinite-depth quadratic correlation taking a distinct value for each bottleneck angle β as well as for each input angle α. This lets us recover information about the inputs from the infinite-depth quadratic correlation, indicating that bottlenecks help information propagate to extreme depths (Prop. 6.12 (b)).

The symmetry breaking discussed above is not apparent in the phase transition of degenerate NNGP kernels noted in the literature; i.e., all information about the inputs are lost in an infinite-depth (no-bottleneck) NNGP in either phase. We see, however, that

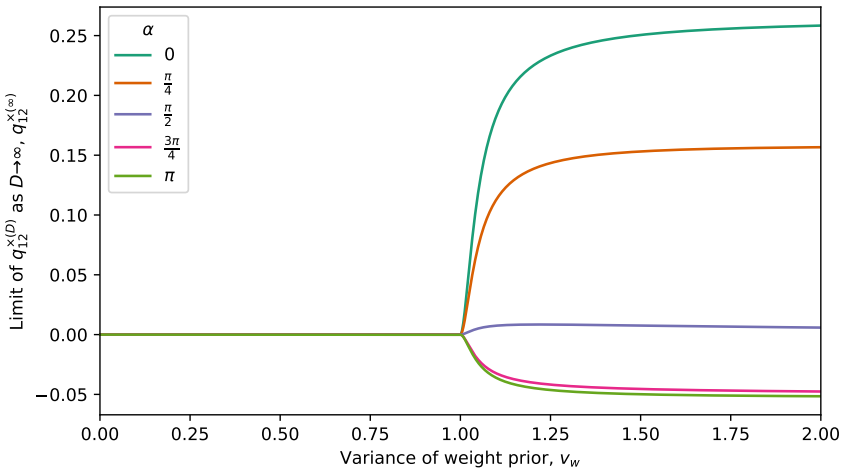


Figure 6.7: The infinite-depth quadratic correlation between outputs as a function of the weight-variance hyperparameter v_w for an example bottleneck NNGP for various angles α between the inputs.

the restriction of just one hidden layer to a bottleneck is sufficient to break this symmetry. Moreover, the symmetry can be recovered by sending the bottleneck width to infinity (Thm. 6.8); the bottleneck is therefore analogous to an external magnetic field in the Ising model.

6.4.3 A divergent depth scale

Schoenholz et al. [130] show that the characteristic depth scale on which the kernel of a (no-bottleneck) NNGP degenerates exponentially to its deep limit diverges at a phase boundary in (v_b, v_w)-space, and they use this result to argue that values of (v_b, v_w) near criticality or "at the edge of chaos" optimize trainability by maximizing the depth to which information can penetrate in an NNGP. We show that the depth scale on which the quadratic correlation $Q^{\times(D)}$ converges to its limit also diverges at the phase boundary $v_w = 1$.

Proposition 6.13 (Characteristic depth scale)**.** *Consider the single-bottleneck NNGP F with normalized ReLU activation from Prop. 6.10, and suppose it is fed two inputs x_1 and x_2 with $\|x_1\| = \|x_2\|$ and bottleneck angle $\beta \neq \frac{\pi}{2}$. Then for all positive $v_w \neq 1$, the quadratic correlation $q_{ab}^{\times(D)}$ is asymptotically exponential in D, meaning that the limit*

$$L = \lim_{D\to\infty} \frac{q_{ab}^{\times(D)} - q_{ab}^{\times(\infty)}}{e^{-\frac{D}{\lambda}}} \tag{6.17}$$

exists and is finite and non-zero for some $\lambda > 0$, which we find to be

$$\lambda = \begin{cases} \ln\left(\frac{1}{v_w^2}\right)^{-1} & \text{if } v_w < 1 \\ \ln(v_w)^{-1} & \text{if } v_w > 1. \end{cases} \tag{6.18}$$

In the case $v_w = 1$, the limit L is infinite for all finite $\lambda > 0$.

Proposition 6.13 excludes the case of orthogonal bottleneck preactivations ($\beta = \frac{\pi}{2}$) since it leads to trivial asymptotic behavior ($q_{ab}^{\times(D)} = 0$ for all D). The quantity λ given in Eq. (6.18) is called the characteristic depth scale and is proportional to the "half-life" of quadratic correlation at large depth. The divergence of the depth scale at $v_w = 1$ gives us another perspective on the phase transition in bottleneck NNGP models (Fig. 6.8). Based on this, we expect optimal values of the v_w hyperparameter to be greater than but close to 1; although $v_w = 1$ gives the smallest decay rate (i.e., largest depth scale) to the infinite-depth quadratic correlation, this limiting value is 0. Larger values of v_w admit non-zero infinite-depth quadratic correlations, but values that are too large lead to fast decay rates (i.e., small depth scales). We hypothesize that this tension between large depth scales (near $v_w = 1$) and non-zero quadratic correlations ($v_w \gg 1$) is the main driving force determining the optimal value of v_w in the $v_w > 1$ phase.

6.4.4 Non-degenerate kernels at extreme depths

The non-trivial dependence of the infinite-depth quadratic correlation $q_{ab}^{\times(\infty)}(\beta)$ on the bottleneck angle β has remarkable implications for the kernel or covariance matrix $K^{(\infty)}$

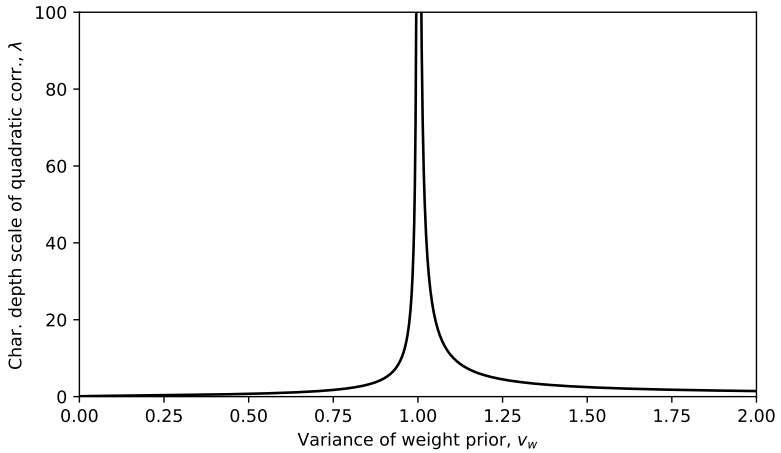


Figure 6.8: Characteristic depth scale of the convergence of the quadratic correlation between outputs to its infinite-depth limit as a function of the weight-variance hyperparameter v_w.

of individual output neurons of a bottleneck NNGP at infinite depth. Consider again the single-bottleneck NNGP F described at the beginning of Sec. 6.4.1, and suppose it is fed two linearly independent inputs x_1 and x_2. Assume that the infinite-depth kernel $K^{(\infty)}$ is degenerate so that the associated correlation matrix $\hat{K}^{(\infty)}$ is a matrix of ones; this is indeed the case when there are no bottlenecks, where the correlations in $\hat{K}$ tend to 100% even when all elements of the kernel K grow to infinity at infinite depth. Then the outputs $F_i(x_1)$ and $F_i(x_2)$ of a single output neuron i at infinite depth are linearly dependent and are in fact equal. It follows that the elements $q_{ab}^{\times(\infty)}$ of the quadratic correlation matrix $Q^{\times(\infty)}$ between distinct output neurons are all equal, but this is impossible in the $v_w > 1$ phase since $q_{12}^{\times(\infty)}(\beta)$ is a one-one function of β (Fig. 6.7) and the diagonal elements assume $\beta = 0$ while the off-diagonal elements assume $\beta > 0$ (since x_1 and x_2 are linearly independent). We therefore learn that the kernel of an NNGP does not degenerate to a constant in the $v_w > 1$ phase if at least one hidden layer is bottlenecked.

Unfortunately, the kernel—and thus the associated correlation matrix—of each output neuron of a bottleneck NNGP do not admit closed forms, even in the infinite depth limit. The quadratic correlation matrix $Q^{(D)}$ (without a superscript $\times$) for individual outputs is intractable as well, but its infinite depth limit does admit an elegant closed form.

Proposition 6.14 (Infinite-depth quadratic correlation for single output)**.** *Consider the single-bottleneck NNGP F with normalized ReLU activation from Prop. 6.10, and suppose we send the post-bottleneck depth to infinity.*

(a) The infinite-depth quadratic correlation matrix $Q^{(\infty)}$ of a single output neuron has (a, b)-th element $q_{ab}^{(\infty)} = \lim_{D\to\infty} q_{ab}^{(D)}$ given by

$$q_{ab}^{(\infty)} = \begin{cases} 3q_{ab}^{\times(\infty)} + \dfrac{\left(\frac{v_b}{(v_w-1)c_{aa}}+1\right)\left(\frac{v_b}{(v_w-1)c_{bb}}+1\right)}{\prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{2H} + \left(\frac{v_b}{(v_w-1)c}+1\right)^2}} & \text{if } v_w > 1 \\ 1 & \text{otherwise.} \end{cases} \tag{6.19}$$

(b) Let G be the Gram matrix of inputs with entries $g_{ab} = x_a^\top x_b$. If $v_w > 1$, then the mapping $G \mapsto (Q^{(\infty)}, \mathrm{diag}(G))$ is invertible.

The infinite-depth single-output quadratic correlation $Q^{(\infty)}$ carries many of the same properties as the infinite-depth between-output quadratic correlation $Q^{\times(\infty)}$. Recalling that $q_{ab}^{\times(\infty)} \to 0$ as $H \to \infty$, it is easy to verify that $q_{ab}^{(\infty)} \to 1$. The single-output quadratic correlation also exhibits the same symmetry breaking at the phase boundary $v_w = 1$ as the between-output quadratic correlation; this is evident in the example plot of $q_{12}^{(\infty)}(\beta)$ as a function of v_w, using the same setup as for Fig. 6.7 (see Fig. 6.9).

In the phase $v_w < 1$, the single-output quadratic correlation is 100%, suggesting degeneracy at infinite depth. In contrast, in the phase $v_w > 1$—where symmetry breaks—$q_{ab}^{(\infty)}$ becomes a strictly increasing function of the input angle α, allowing us to recover some information about the inputs (Prop. 6.14 (b)). In particular, Prop. 6.14 (b) implies that in the ordered phase $v_w > 1$, if the norms of the inputs are known (if the inputs are constrained to a

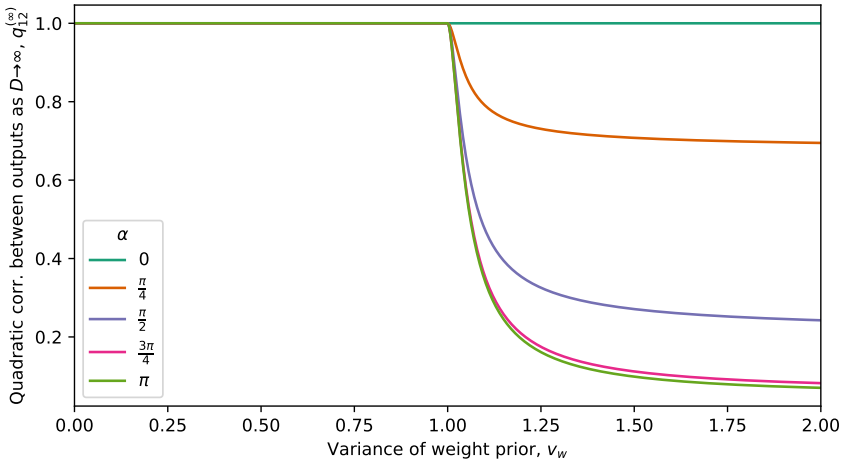


Figure 6.9: The infinite-depth quadratic correlation of a single output as a function of the v_w hyperparameter for an example bottleneck NNGP.

sphere, for example), then the input angle can be recovered from the single-output quadratic correlation even at infinite depth. This stands in stark contrast to no-bottleneck NNGP models, where all information is lost at infinite depth regardless of the phase, and it suggests that bottleneck layers are vital for the trainability of very deep models.

6.4.5 Other nonlinearities

The behavior described in Prop. 6.14 does not extend to other common nonlinearities and is thus all the more striking. Two notable examples are the sigmoidal nonlinearity $\phi(x) = \tanh x$ and the sinusoidal nonlinearity

$$\phi(x) = \cos x + \sin x. \tag{6.20}$$

The significance of the latter is that the corresponding NNGP has an RBF kernel; the random features literature [123, 33] hints at this connection but does not discuss it in the context of NNGPs or neural network nonlinearities. We make this connection more precise in the next proposition and subsequent remark (see Appendix I.1 for the proofs of both Props. 6.15, 6.17).

Proposition 6.15 (RBF-NNGP kernel recursion)**.** *Consider an NNGP mapping $\mathbb{R}^M$ to $\mathbb{R}$ with D hidden layers and the sinusoidal nonlinearity in Eq. (6.20).*

(a) The NNGP kernel $K^{(\mu)} : \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}$ for the μ-th hidden layer is given recursively as

$$K^{(1)}(x, x') = v_b + v_w x^\top x' \tag{6.21}$$

$$K^{(\mu+1)}(x, x') = v_b + v_w e^{-\frac{1}{2}\left[K^{(\mu)}(x,x)+K^{(\mu)}(x',x')-2K^{(\mu)}(x,x')\right]}, \tag{6.22}$$

for $\mu = 1, \ldots, D$ where $K^{(D+1)}$ is the kernel of the output layer.

(b) In the deep limit $D \to \infty$, the NNGP kernel converges pointwise to

$$K^{(\infty)}(x, x') = v_*(v_b, v_w) \begin{cases} 1, & \text{if } x = x' \\ c_*(v_b, v_w), & \text{otherwise,} \end{cases} \tag{6.23}$$

where

$$v_*(v_b, v_w) = v_b + v_w \tag{6.24}$$

$$c_*(v_b, v_w) = \begin{cases} 1, & \text{if } v_w < 1 \\ c', & \text{if } v_w > 1. \end{cases} \tag{6.25}$$

Remark 6.16 (RBF-DGP as a bottleneck NNGP)**.** *Since the sinusoidal nonlinearity in Eq. (6.20) clearly satisfies the linear envelope condition, then we can apply Thm. 6.4 to a BNN with the sinusoidal nonlinearity and obtain a bottleneck NNGP limit. If no bottlenecks are*

imposed, then the wide limit of such a BNN is the NNGP described in Prop. 6.15. In particular, using Eqs. (6.21)-(6.22), the single-hidden-layer NNGP with sinusoidal nonlinearity has kernel

$$\begin{aligned} K^{(2)}(x,x') &= v_b + v_w e^{-\frac{1}{2}[K^{(1)}(x,x)+K^{(1)}(x',x')-2K^{(1)}(x,x')]} \\ &= v_b + v_w e^{-\frac{1}{2}[v_b+v_w\|x\|^2+v_b+v_w\|x'\|^2-2(v_b+v_w x\cdot x')]} \\ &= v_b + v_w e^{-\frac{v_w}{2}\|x-x'\|^2}, \end{aligned}$$

which we recognize as the RBF kernel. More generally, if the BNN with sinusoidal nonlinearity has an odd number of hidden layers D and the μ-th hidden layers are restricted to bottlenecks for $\mu = 2, 4, 6, \ldots, D-1$, then the bottleneck NNGP limit is a DGP with $\frac{D+1}{2}$ GP components each with RBF kernel (and sinusoidal nonlinearities applied to the bottleneck layers).

The deep limit of an NNGP with sinusoidal nonlinearity is described in Prop. 6.15 (b). It is identical to the behavior of an NNGP with sigmoidal nonlinearity as described by Poole et al. [121], except that $v_*(v_b, v_w)$ and $c_*(v_b, v_w)$ take different forms and the phase boundary has a different location. In one phase ($v_w < 1$ for the sinusoidal nonlinearity), all inputs tend to 100% correlation as depth is increased without bound—similar to the ReLU activation. However, in the other phase ($v_w > 1$ for the sinusoidal nonlinearity)—the "chaotic phase"—the infinite-depth correlation for distinct inputs is a constant less than 100%; thus, unlike the ReLU activation, the sigmoidal and sinusoidal nonlinearities allow distinct inputs to remain distinct through infinite depth, although all information about the distince between distinct inputs is lost. Moreover, the introduction of a bottleneck does not remove this degeneracy—in sharp contrast to Prop. 6.14. We substantiate this with the next proposition.

Proposition 6.17 (Deep post-bottleneck limit for sigmoidal and sinusoidal nonlinearities)**.** *Consider a single-bottleneck NNGP F mapping $\mathbb{R}^M$ to $\mathbb{R}$ with either sigmoidal or sinusoidal nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$. Suppose IID Gaussian noise $\mathcal{N}(0, v_n)$ is added to the outputs for any $v_n > 0$. Given two distinct inputs $x_1, x_2 \in \mathbb{R}^M$, let $p^{(D)} : \mathbb{R}^2 \mapsto [0, \infty)$ be the PDF of $\{F(x_1), F(x_2)\}$, where D is the post-bottleneck depth. Then in the deep post-bottleneck limit $D \to \infty$, the PDF converges pointwise to*

$$p^{(\infty)}(y) = \mathcal{N}(y; 0, K^{(\infty)}(X, X) + v_n I), \tag{6.26}$$

where $K^{(\infty)}(X, X)$ is a 2×2 matrix with entries $K^{(\infty)}(x_a, x_b)$ with $K^{(\infty)}$ given in Eq. (6.23).

Note that Eq. (6.26) is independent of bottleneck width; in fact, it is exactly the PDF of a no-bottleneck NNGP, indicating that the bottleneck has no effect at all on the deep post-bottleneck limit given either the sigmoidal or sinusoidal nonlinearity. In contrast to Prop. 6.14 (b) for the ReLU activation, the only information about the inputs that can be recovered from the deep post-bottleneck limit is whether the inputs are distinct. This suggests a fundamental difference between the ReLU activation and the sigmoidal nonlinearity and between the NNGP kernel with ReLU activation and the RBF kernel—namely, that the ReLU activation allows a network to operate at very large depths as long as one bottleneck is present. Classifying activation functions based on their deep limit and deep post-bottleneck limit behaviors could help to better understand which activation functions are useful in practice and is a topic for future work.

6.4.6 Other deep limits

Besides the deep post-bottleneck limit discussed in Secs. 6.4.2-6.4.4, there are two other limits we could have considered: 1) The deep pre-bottleneck limit where a single-bottleneck NNGP has infinitely many infinitely wide hidden layers before its bottleneck but only finitely many after, and 2) the doubly deep bottleneck limit where the single-bottleneck NNGP has infinitely many infinitely wide hidden layers both before and after its bottleneck. The quadratic correlation between the activations of distinct output neurons can be analyzed in both of these limits by replacing the pre-bottleneck NNGP covariance matrix C in Prop. (6.10) with a sequence of such covariance matrices indexed by the pre-bottleneck depth. Since the NNGP kernel with ReLU activation tends to a constant in the deep limit, then the bottleneck angle β tends to 0 in both the deep pre-bottleneck limit and doubly deep bottleneck limit. Although the quadratic correlation remains nontrivial in both limits and even exhibits a first-order phase transition at $v_w = 1$ in the doubly deep bottleneck limit, these results are uninteresting as all information about the original inputs into the network is lost due to vanishing bottleneck angle.

The quadratic correlation between the activations of a single output neuron at two different inputs can also be analyzed in both the deep pre-bottleneck and doubly deep bottleneck limits; calculations proceed similarly as in the proof of Prop. 6.14. Again, since the bottleneck angle tends to 0, we find that the quadratic correlation tends to 100% in both limits at every post-bottleneck depth; the bottleneck NNGP kernel therefore degenerates, unlike in the deep post-bottleneck limit. We conclude that out of the three possible deep limits, only the deep post-bottleneck limit is interesting since it is the only one that admits a phase ($v_w > 1$) in which information about network inputs is preserved.

6.5 Discussion

Our main theorem, Thm. 6.4, generalizes the result of Matthews et al., 2018 concerning deep neural networks whose hidden layer widths are increased without bound to a setting in which some intermediate hidden layers, called bottlenecks, are fixed to a finite width. From a theoretical perspective, this result connects the NNGP literature with that of DGPs, as the resulting probability model is in fact a DGP consisting of a composition of NNGP components. Additionally, we have explored the effect of these bottleneck layers on the resulting probability model from a practical perspective, showing that model likelihood peaks at a finite bottleneck width and is superior to that of no-bottleneck NNGPs.

Surprisingly, in contrast to no-bottleneck NNGP models, the behavior of a bottleneck NNGP with ReLU activation at extreme post-bottleneck depths is not always degenerate (Props. 6.12, 6.14); in particular, the input Gram matrix can be fully recovered from the between-output quadratic correlation matrix of the bottleneck NNGP at infinite post-bottleneck depth, and the input angle can be recovered even from the single-output quadratic correlation matrix at infinite depth if the input norms are known. Bottleneck layers are therefore fundamental as they allow networks to "go deeper". However, this non-degeneracy in the deep limit manifests only when the network weight prior is weaker than a standard normal. We have just begun to explore the dependence of the deep post-bottleneck limit on

the prior weight variance, showing that convergence to the limit is asymptotically exponential in depth and that the characteristic depth scale diverges at a critical value of the weight prior variance.

So far, we have not directly connected bottleneck NNGPs with BNNs, aside from studying the limits of BNNs in the wide regime. However, an interesting special case of bottleneck NNGPs emerges when *every* hidden layer is fixed at finite width. The result is a BNN, but our result lends an interesting perspective that suggests that one might approach BNN inference with a non-parametric DGP-based approach. Indeed, in follow-up work we intend to explore the practicality of such a method. Moreover, we believe that our main theorem can easily be extended to convolutional architectures (by introducing a second index for each hidden layer), and thus we plan to explore the implications of our work for convolutional BNNs.

Finally, in this work we did not consider the implications of the bottleneck NNGP for the learning dynamics of Gaussian-initialized deterministic neural networks (DNNs). It is now a celebrated result that the evolution over training time of a Gaussian-initialized DNN is described in the wide limit by an exactly solvable linear ODE, where the time evolution operator is termed the neural tangent kernel (NTK) and is related to the NNGP kernel [75]. As part of additional follow-up work, we plan to investigate the "bottleneck NTK", where the wide limit is relaxed to allow for some bottleneck layers. The result is a system of coupled ODEs that is more challenging to analyze but carries potential for a more refined description of the evolution of finite-width DNNs.

Chapter 7

Conclusion

The works in this thesis show that Bayesian DL can be beneficial for robust scientific data analysis and is rich with new classes of methods that promise practical benefit in future applications. Chapters 3-4 show that DKL and BNNs are beneficial for text classification, with BNNs now the new state-of-the-art method for classifying cancer pathology reports. The DKL work also demonstrates the effectiveness of the linear kernel in the final layer, in contrast to the more popular RBF kernel. We speculate that this is because the linear kernel is non-local, which prevents it from degenerating in high dimensions due to the curse of dimensionality [14]. Future directions for the DKL and BNN works focus on practical issues of trainability and pretraining for DKL as well as the development of a Bayesian HiSAN model with Bayesian word2vec, which could improve predictive performance on the cancerpathology reports classification problem even further.

Chapter 5 shows the benefits of DEKL– a hybrid of DKL and deep ensembles. DEKL allows us to parallelize and thus scale the DKL paradigm, which could be critical to tackling more complex scientific problems. As additional future work, we plan to investigate the connection between posterior variance in DKL and diversity in deep ensembles– a key question at the intersection of GP and DL methods. understanding this connection could help us explain the benefit of DEKL and could offer insight into how we could improve it further.

Finally, in Chapter 6, we introduce the bottleneck NNGP, which we see as a foundation or springboard for both theoretical and practical work in the future. For example, the no-bottleneck NNGP has been used to understand the learning dynamics of NNs and to obtain generalization bounds; could we refine these results with bottlenecks, obtaining tighter generalization bounds and a more accurate description of NN dynamics? More practically, can we characterize the datasets or problems for which the bottleneck NNGP is a better model than the traditional DGP with RBF kernels? In previous work [2], we found that a no-bottleneck NNGP outperforms an RBF-GP for thoracic disease detection on the ChestXRay14 dataset [145]. As with the linear kernel, we surmise that this is because the NNGP kernel is non-local and thus does not degenerate in high dimensions (each x-ray in the dataset is 1024×1024). These works therefore support the hypothesis that DKL with a linear base kernel or a DGP with NNGP kernels (i.e., a bottleneck NNGP) may be more suitable for high-dimensional scientific data analysis than traditional GP or kernel methods.

The experiments in this thesis are limited to supervised learning problems, but the methods we have presented could also be useful in other learning paradigms– perhaps more so. For example, we could use our methods for active learning, where a model identifies regions in input space where new labeled training examples would be most beneficial to boost model performance. We could interpret high predictive variance of a DKL model or the bottleneck NNGP, for instance, as indication that a region of input space needs additional exploration. Active learning is relevant to the problem of automated information extraction from cancer pathology reports [78]; since annotation of reports by human experts is expensive, developing a model that can identify which reports should be manually annotated next and added to the training corpus would be a significant contribution. Similar to active learning, we could also apply our methods to Bayesian optimization and surrogate modeling, which are highly relevant in the computational sciences and where GP models are ubiquitous [131]. DKL may be particularly well-suited as it will embed the input space—which is typically discrete (e.g., space of model hyperparameters)—into a Euclidean space where relationships between

inputs (e.g., different hyperparameter combinations) may be encoded, analogous to word embeddings in NLP.

As a final future direction, although Bayesian methods are typically beneficial on smaller datasets requiring more robust models, Bayesian DL may ironically hold the key to large-scale distributed DL. We have already mentioned the scalability of DEKL as a topic for future work, but more generally, Bayesian DL may allow us to perform SGD with larger minibatches and thus greater data parallelism, while maintaining the stochasticity needed for robust training. Indeed, Bayesian DL is not just about limited data and UQ but is more generally about trainability and generalizability– two of the most important topics in DL.

The future directions outlined above only serve to showcase the broad applicability of Bayesian DL, covering multiple priority research directions laid out in the 2019 OSTI report on SciML. As this thesis demonstrates, the cross-pollenation of GP and DL methods is not only effective but is rich with possibilities for new methods and foundational insight that may help to advance robust SciML even further going forward.

Bibliography

[1] Agrawal, D., Papamarkou, T., and Hinkle, J. (2020). Wide neural networks with bottlenecks are deep Gaussian processes. *Journal of Machine Learning Research*, 21(175):1–66. 55, 70

[2] Agrawal, D., Yoon, H.-J., Tourassi, G., and Hinkle, J. D. (2019). Computer-aided detection using non-convolutional neural network Gaussian processes. In *Medical Imaging 2019: Computer-Aided Diagnosis*, volume 10950, page 109503N. International Society for Optics and Photonics. 102

[3] Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. (2017). Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research*, 18(82):1–37. 11, 19, 55

[4] Alaa, A. M. and van der Schaar, M. (2017). Deep multi-task Gaussian processes for survival analysis with competing risks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2326–2334. Curran Associates Inc. 12, 73, 86

[5] Alawad, M., Gao, S., Qiu, J. X., Yoon, H. J., Blair Christian, J., Penberthy, L., Mumphrey, B., Wu, X.-C., Coyle, L., and Tourassi, G. (2020). Automatic extraction of cancer registry reportable information from free-text pathology reports using multitask convolutional neural networks. *Journal of the American Medical Informatics Association*, 27(1):89–98. 15

[6] Alawad, M., Gao, S., Qiu, J. X., Yoon, H.-J., Christian, J. B., Penberthy, L., Mumphrey, B., Wu, X.-C., Coyle, L., and Tourassi, G. (2019). Automatic extraction of cancer registry reportable information from free-text pathology reports using multi-task convolutional neural networks. *Journal of the American Medical Informatics Association*. to appear. 19, 39, 43, 44

[7] Alvarez, M. and Lawrence, N. D. (2009). Sparse convolved Gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pages 57–64. 73

[8] Anderson, E. (1935). The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59:2–5. 80

[9] Antipov, G., Berrani, S.-A., Ruchaud, N., and Dugelay, J.-L. (2015). Learned vs. hand-crafted features for pedestrian gender recognition. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1263–1266. ACM. 2, 70

[10] Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8139–8148. 75

[11] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. 46

[12] Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., Parashar, M., Patra, A., Sethian, J., Wild, S., Willcox, K., and Lee, S. (2019). Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. 2

[13] Baxter, R. J. (2016). *Exactly solved models in statistical mechanics*. Elsevier. 91

[14] Bengio, Y., Delalleau, O., and Le Roux, N. (2005). The curse of dimensionality for local kernel machines. *Techn. Rep*, 1258. 10, 25, 102

[15] Billingsley, P. (1995). *Probability and measure*. Wiley Series in Probability and Statistics: Probability and Statistics. John Wiley & Sons Inc., New York, third edition. 136

[16] Billingsley, P. (1999). *Convergence of probability measures*. Wiley Series in Probability and Statistics: Probability and Statistics. John Wiley & Sons Inc., New York, second edition. 130, 143, 144

[17] Blomqvist, K., Kaski, S., and Heinonen, M. (2018). Deep convolutional Gaussian processes. *arXiv preprint arXiv:1810.03052*. 13

[18] Blum, J. R., Chernoff, H., Rosenblatt, M., and Teicher, H. (1958). Central limit theorems for interchangeable processes. *Canadian Journal of Mathematics*, 10. 75, 142

[19] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France. PMLR. 8, 43, 46

[20] Bonilla, E. V., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160. 12, 73, 84

[21] Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. (2017). Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*. 11, 19, 27, 28, 31, 55

[22] Brown, G., Wyatt, J. L., and Tiňo, P. (2005). Managing diversity in regression ensembles. *Journal of machine learning research*, 6(Sep):1621–1650. 66

[23] Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. (2016). Deep Gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pages 1472–1481. 12, 73

[24] Bullins, B., Zhang, C., and Zhang, Y. (2018). Not-so-random features. In *International Conference on Learning Representations*. 55

[25] Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. (2016). Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE. 10

[26] Carrell, D. S., Halgrim, S., Tran, D. T., Buist, D. S. M., Chubak, J., Chapman, W. W., and Savova, G. (2014). Using natural language processing to improve efficiency of manual chart abstraction in research: the case of breast cancer recurrence. *Am J Epidemiol*, 179(6):749–758. 15

[27] Chen, A. M. and Hecht-Nielsen, R. (1991). On the geometry of feedforward neural network weight spaces. In *1991 Second International Conference on Artificial Neural Networks*, pages 1–4. IET. 9

[28] Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350. 13, 74, 79, 82, 85, 138, 153

[29] Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46. 29

[30] Coden, A., Savova, G., Sominsky, I., Tanenblatt, M., Masanz, J., Schuler, K., Cooper, J., Guan, W., and De Groen, P. C. (2009). Automatically extracting cancer disease characteristics from pathology reports into a disease knowledge representation model. *Journal of biomedical informatics*, 42(5):937–949. 15

[31] Connolly, J. L., Schnitt, S. J., Wang, H. H., Longtine, J. A., Dvorak, A., and Dvorak, H. F. (2003). Role of the surgical pathologist in the diagnosis and management of the cancer patient. 15

[32] Cuong, H. and Xu, J. (2018). Assessing quality estimation models for sentence-level prediction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1521–1533. 11, 20

[33] Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. (2017). Random feature expansions for deep Gaussian processes. In *International Conference on Machine Learning*, pages 884–893. 68, 97

[34] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314. 58

[35] Dai, Z., Damianou, A., González, J., and Lawrence, N. (2015). Variational auto-encoded deep Gaussian processes. *arXiv preprint arXiv:1511.06455*. 12, 73

[36] Damianou, A. and Lawrence, N. (2013). Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215. 12, 55, 68, 71, 73, 85

[37] Daniely, A., Frostig, R., and Singer, Y. (2016). Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pages 2253–2261. 13, 75

[38] Dasgupta, S., Sricharan, K., and Srivastava, A. (2018). Finite rank deep kernel learning. In *Third Workshop on Bayesian Deep Learning, NeurIPS*. 55, 68

[39] de G. Matthews, A. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6. 10, 27, 28

[40] de Matthews, A. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *International Conference on Learning Representations*. accepted as poster. 13, 55, 72, 73, 75, 76, 77, 79, 99, 130, 132, 133, 139, 141, 142, 143, 144, 145, 146, 147, 148

[41] Dua, D. and Graff, C. (2017). UCI machine learning repository. 21

[42] Dubey, A. K., Hinkle, J., Christian, J. B., and Tourassi, G. (2019). Extraction of tumor site from cancer pathology reports using deep filters. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB '19, pages 320–327, New York, NY, USA. ACM. 21

[43] Dunlop, M. M., Girolami, M. A., Stuart, A. M., and Teckentrup, A. L. (2018). How deep are deep Gaussian processes? *The Journal of Machine Learning Research*, 19(1):2100–2145. 12, 74

[44] Dutordoir, V., Salimbeni, H., Hensman, J., and Deisenroth, M. (2018). Gaussian process conditional density estimation. In *Advances in neural information processing systems*, pages 2385–2395. 73

[45] Duvenaud, D., Rippel, O., Adams, R., and Ghahramani, Z. (2014). Avoiding pathologies in very deep networks. In *Artificial Intelligence and Statistics*, pages 202–210. 12, 74

[46] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188. 80

[47] Föll, R., Haasdonk, B., Hanselmann, M., and Ulmer, H. (2017). Deep recurrent Gaussian process with variational sparse spectrum approximation. *arXiv preprint arXiv:1711.00799*. 13

[48] Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian approximation: representing model uncertainty in deep learning. *ArXiv preprint arXiv:1506.02142*. 8, 12, 20, 74

[49] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. 54, 56

[50] Gal, Y., Van Der Wilk, M., and Rasmussen, C. E. (2014). Distributed variational inference in sparse Gaussian process regression and latent variable models. In *Advances in neural information processing systems*, pages 3257–3265. 10

[51] Gao, S., Qiu, J. X., Alawad, M., Hinkle, J. D., Schaefferkoetter, N., Yoon, H.-J., Christian, B., Fearn, P. A., Penberthy, L., Wu, X.-C., et al. (2019). Classifying cancer pathology reports with hierarchical self-attention networks. *Artificial Intelligence in Medicine*, 101:101726. 15, 19, 39, 43, 44, 46, 48

[52] Gao, S., Young, M. T., Qiu, J. X., Yoon, H.-J., Christian, J. B., Fearn, P. A., Tourassi, G. D., and Ramanthan, A. (2017). Hierarchical attention networks for information extraction from cancer pathology reports. *Journal of the American Medical Informatics Association*, 25(3):321–330. 15, 19, 39, 43

[53] Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586. 10

[54] Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. (2019). Deep convolutional networks as shallow Gaussian processes. In *International Conference on Learning Representations*. 14, 75

[55] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press. 4

[56] Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356. 8, 43, 46

[57] Guhaniyogi, R. and Dunson, D. B. (2016). Compressed Gaussian process for manifold regression. *The Journal of Machine Learning Research*, 17(1):2472–2497. 10

[58] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org. 19, 54

[59] Hansen, L. K., Liisberg, C., and Salamon, P. (1997). The error-reject tradeoff. *Open Systems & Information Dynamics*, 4(2):159–184. 29

[60] Harrison, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102. 80

[61] Havasi, M., Hernández-Lobato, J. M., and Murillo-Fuentes, J. J. (2018). Inference in deep Gaussian processes using stochastic gradient hamiltonian monte carlo. In *Advances in Neural Information Processing Systems*, pages 7506–7516. 12

[62] Hazan, T. and Jaakkola, T. (2015). Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*. 13, 75

[63] Hecht-Nielsen, R. (1990). On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier. 8, 9

[64] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*. 10, 12, 26, 60, 73

[65] Hensman, J. and Lawrence, N. D. (2014). Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*. 12, 73

[66] Hensman, J., Matthews, A., and Ghahramani, Z. (2015a). Scalable variational Gaussian process classification. 10

[67] Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. (2015b). MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1648–1656. 27

[68] Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869. 56, 63

[69] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507. 14, 75

[70] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. 8

[71] Hinton, G. E. and Van Camp, D. (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*, pages 5–13. 8

[72] Hornik, K., Stinchcombe, M., White, H., et al. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366. 58

[73] Izmailov, P., Novikov, A., and Kropotov, D. (2017). Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition. *arXiv preprint arXiv:1710.07324*. 10, 11, 19

[74] Izmailov, P., Novikov, A., and Kropotov, D. (2018). Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition. In *International Conference on Artificial Intelligence and Statistics*, pages 726–735. 55

[75] Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580. 14, 54, 55, 75, 100

[76] Jean, N., Xie, M., and Ermon, S. (2016). Semi-supervised deep kernel learning. In *NIPS Bayesian deep learning workshop*. 11, 19, 55

[77] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584. 20, 54

[78] Kholghi, M., Sitbon, L., Zuccon, G., and Nguyen, A. (2016). Active learning: a step towards automating medical concept extraction. *Journal of the American Medical Informatics Association*, 23(2):289–296. 102

[79] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–175, Doha, Qatar. Association for Computational Linguistics. 15, 19, 20, 21, 28, 44

[80] Kingma, D. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*. 14, 75

[81] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 82

[82] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583. 8, 46, 54, 82

[83] Krishnan, R., Subedar, M., and Tickoo, O. (2019). Efficient priors for scalable variational inference in Bayesian deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0. 52

[84] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 2, 70

[85] Kumar, V., Singh, V., Srijith, P., and Damianou, A. (2018). Deep Gaussian processes with convolutional kernels. *arXiv preprint arXiv:1806.01655*. 13

[86] Kurková, V. and Kainen, P. C. (1994). Functionally equivalent feedforward neural networks. *Neural Computation*, 6(3):543–558. 9

[87] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413. 55, 58, 61, 63

[88] Lawrence, N. D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in neural information processing systems*, pages 329–336. 73

[89] Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as Gaussian processes. *arXiv preprint arXiv:1711.00165*. 13, 14, 19, 72, 75, 80, 82, 91

[90] Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018a). Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*. 55

[91] Lee, J., Song, H.-J., Yoon, E., Park, S.-B., Park, S.-H., Seo, J.-W., Park, P., and Choi, J. (2018b). Automated extraction of biomarker information from pathology reports. *BMC medical informatics and decision making*, 18(1):29. 15

[92] Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*. 14, 75

[93] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867. 58

[94] Li, C.-L., Chang, W.-C., Mroueh, Y., Yang, Y., and Poczos, B. (2019). Implicit kernel learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2007–2016. 55

[95] Li, P. and Nguyen, P.-M. (2019). On random deep weight-tied autoencoders: Exact asymptotic analysis, phase transitions, and implications to training. In *International Conference on Learning Representations*. 75

[96] Li, Y. and Martinez, D. (2010). Information extraction of multiple categories from pathology reports. *Proc ALTA*, pages 41–48. 15

[97] Liang, H., Sun, X., Sun, Y., and Gao, Y. (2017). Text feature extraction based on deep learning: a review. *EURASIP journal on wireless communications and networking*, 2017(1):211. 2, 70

[98] Lin, Z., Memisevic, R., and Konda, K. (2015). How far can we go without convolution: Improving fully-connected networks. *arXiv preprint arXiv:1511.02580*. 75

[99] Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2020). When Gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems*. 10

[100] Louizos, C. and Welling, M. (2016). Structured and efficient variational deep learning with matrix Gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716. 8, 52

[101] Lu, C.-K., Yang, S. C.-H., Hao, X., and Shafto, P. (2019). Interpretable deep Gaussian processes with moments. *arXiv preprint arXiv:1905.10963*. 12, 74

[102] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics. 21

[103] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605. 32

[104] Martinez, D. and Li, Y. (2011). Information extraction from pathology reports in a hospital setting. In *Proc ACM Int Conf Inf Knowl Manag*, pages 1877–1882. ACM. 15

[105] Mattos, C. L. C. and Barreto, G. A. (2019). A stochastic variational framework for recurrent Gaussian processes models. *Neural Networks*, 112:54–72. 13

[106] Mattos, C. L. C., Dai, Z., Damianou, A., Barreto, G. A., and Lawrence, N. D. (2017). Deep recurrent Gaussian processes for outlier-robust system identification. *Journal of Process Control*, 60:82–94. 13

[107] Mattos, C. L. C., Dai, Z., Damianou, A., Forth, J., Barreto, G. A., and Lawrence, N. D. (2015). Recurrent Gaussian processes. *arXiv preprint arXiv:1511.06644*. 13

[108] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446. 58, 122

[109] Meyer, P. A. (1966). *Probability and potentials*, volume 1318. Blaisdell Pub. Co. 148

[110] Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667. 58

[111] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. 14, 75

[112] Nadeem, M. S. A., Zucker, J.-D., and Hanczar, B. (2009). Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In Dşeroski, S., Guerts, P., and Rousu, J., editors, *Proceedings of the third International Workshop on Machine Learning in Systems Biology*, volume 8 of *Proceedings of Machine Learning Research*, pages 65–81, Ljubljana, Slovenia. PMLR. 29

[113] Neal, R. M. (1996). Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer. 13, 19, 55, 70, 72, 74, 75

[114] Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media. 4, 20, 43, 54

[115] Nguyen, A. N., Lawley, M. J., Hansen, D. P., Bowman, R. V., Clarke, B. E., Duhig, E. E., and Colquist, S. (2010). Symbolic rule-based classification of lung cancer stages from free-text pathology reports. *J Am Med Inform Assoc*, 17(4):440–445. 15

[116] Nguyen, A. N., Moore, J., O'Dwyer, J., and Philpot, S. (2015). Assessing the utility of automatic cancer registry notifications data extraction from free-text pathology reports. In *AMIA Annual Symposium Proceedings*, volume 2015, page 953. American Medical Informatics Association. 15

[117] Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. (2019). Bayesian deep convolutional networks with many channels are Gaussian processes. In *International Conference on Learning Representations*. 14, 75

[118] Peterson, C. (1987). A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019. 8

[119] Pinkus, A. (1999). Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195. 58, 122

[120] Pinto, A. M., Oliveira, H. G., and Alves, A. O. (2016). Comparing the performance of different NLP toolkits in formal and social media text. In *Proc SLATE*, volume 51, page 16. 15

[121] Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. (2016). Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368. 13, 14, 75, 91, 98

[122] Qiu, J. X., Yoon, H.-J., Fearn, P. A., and Tourassi, G. D. (2017). Deep learning for automated extraction of primary sites from cancer pathology reports. *IEEE journal of biomedical and health informatics*, 22(1):244–251. 15, 16, 19, 20, 21, 23, 43, 44

[123] Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184. 97

[124] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. 54

[125] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA. 4, 9, 19, 74

[126] Rosenblat, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408. 78

[127] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*. 12, 73

[128] Saikkonen, P. (1993). Continuous weak convergence and stochastic equicontinuity results for integrated processes with an application to the estimation of a regression model. *Econometric Theory*, 9(2):155–188. 131

[129] Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4588–4599. 12, 55, 68, 73, 85

[130] Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. (2016). Deep information propagation. *arXiv preprint arXiv:1611.01232*. 13, 14, 54, 75, 91, 93

[131] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175. 102

[132] Sinha, A. and Duchi, J. C. (2016). Learning kernels with random features. In *Advances in Neural Information Processing Systems*, pages 1298–1306. 55

[133] Smith, S. L. and Le, Q. V. (2018). A Bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*. 54

[134] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. 8

[135] Sun, S., Chen, C., and Carin, L. (2017). Learning structured weight uncertainty in Bayesian neural networks. In *Artificial Intelligence and Statistics*, pages 1283–1292. 8

[136] Sun, S., Zhang, G., Shi, J., and Grosse, R. B. (2019). Functional variational Bayesian neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. 9

[137] Sweeting, T. (1980). Uniform asymptotic normality of the maximum likelihood estimator. *The Annals of Statistics*, pages 1375–1381. 130, 131

[138] Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE. 14, 75

[139] Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574. 10, 26, 60, 73

[140] Titsias, M. and Lawrence, N. D. (2010). Bayesian Gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 844–851. 73

[141] Tran, G.-L., Bonilla, E. V., Cunningham, J., Michiardi, P., and Filippone, M. (2019). Calibrating deep convolutional Gaussian processes. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1554–1563. PMLR. 13

[142] Van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 2849–2858. 13

[143] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. 46

[144] Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact Gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pages 14622–14632. 10

[145] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. (2017). ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. *CoRR*, abs/1705.02315. 102

[146] Wang, Y., Brubaker, M., Chaib-Draa, B., and Urtasun, R. (2016). Sequential inference for deep Gaussian process. In *Artificial Intelligence and Statistics*, pages 694–703. 12, 73, 86

[147] Wang, Y., Wang, L., Rastegar-Mojarad, M., Moon, S., Shen, F., Afzal, N., Liu, S., Zeng, Y., Mehrabi, S., Sohn, S., and Liu, H. D. (2018). Clinical information extraction applications: A literature review. *Journal of Biomedical Informatics*, 77:34–49. 15

[148] Webb, A. M., Reynolds, C., Chen, W., Reeve, H., Iliescu, D.-A., Lujan, M., and Brown, G. (2019). To Ensemble or Not Ensemble: When does End-To-End Training Fail? *arXiv preprint arXiv:1902.04422*. 56

[149] Weegar, R. and Dalianis, H. (2015). Creating a rule based system for text mining of norwegian breast cancer pathology reports. In *Proc LOUHI*, pages 73–78. 15

[150] Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*. 8, 43, 47

[151] Williams, C. K. (1997). Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301. 12, 13, 19, 74

[152] Wilson, A. and Nickisch, H. (2015). Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pages 1775–1784. 10, 28

[153] Wilson, A. G. (2020). The case for Bayesian deep learning. *arXiv preprint arXiv:2001.10995*. 54

[154] Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016a). Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378. 10, 11, 19, 27, 55, 56

[155] Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016b). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594. 11, 19, 27, 28, 31, 55, 56

[156] Xie, F., Lee, J., Munoz-Plaza, C. E., Hahn, E. E., and Chen, W. (2017). Application of text information extraction system for real-time cancer case identification in an integrated healthcare organization. *Journal of pathology informatics*, 8. 15

[157] Xuan, J., Lu, J., Yan, Z., and Zhang, G. (2018). Bayesian deep reinforcement learning via deep kernel learning. *International Journal of Computational Intelligence Systems*, 12(1):164–171. 11, 19, 55

[158] Yala, A., Barzilay, R., Salama, L., Griffin, M., Sollender, G., Bardia, A., Lehman, C., Buckley, J. M., Coopey, S. B., Polubriaginof, F., et al. (2017). Using machine learning to parse breast pathology reports. *Breast cancer research and treatment*, 161(2):203–211. 15

[159] Yang, G. (2019a). Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*. 14, 75

[160] Yang, G. (2019b). Tensor programs I: Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. *arXiv preprint arXiv:1910.12478*. 14

[161] Yang, G. and Schoenholz, S. (2017). Mean field residual networks: on the edge of chaos. In *Advances in neural information processing systems*, pages 7103–7114. 14, 75

[162] Yang, Y., Dunson, D. B., et al. (2016a). Bayesian manifold regression. *The Annals of Statistics*, 44(2):876–905. 10

[163] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016b). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489. 19, 46

[164] Yim, W.-w., Denman, T., Kwan, S. W., and Yetisgen, M. (2016). Tumor information extraction in radiology reports for hepatocellular carcinoma patients. *AMIA Summits on Translational Science Proceedings*, 2016:455. 15

[165] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*. 15

[166] Zheng, S., Lu, J. J., Appin, C., Brat, D., and Wang, F. (2015). Support patient search on pathology reports with interactive online learning based data extraction. *Journal of pathology informatics*, 6. 15

[167] Zhou, Y., Chen, C., and Xu, J. (2019). KernelNet: A data-dependent kernel parameterization for deep generative modeling. *arXiv preprint arXiv:1912.00979*. 55

Appendices

A Results for additional DKL models

Here we list the results for the CNN-2 and DKL-RBF models and their pretrained variants (Tables 1-2). These models have more complex classifiers than the CNN-1 and DKL-lin models. The DKL-RBF model often outperforms the CNN-2 baselines, though pretraining appears more critical when using an RBF kernel instead of a linear kernel. These models cannot match the performances of the CNN-1 and DKL-lin models, indicating that more complex classifiers are not necessary for the tasks at hand. Moreover, it indicates that the success of the DKL-lin model is not solely due to its more flexible nonparametric classifier at test time.

Table 1: Mean test F_{micro} scores (as percentages) with standard deviations for each model and dataset. A score is in bold if the mean performance of the model is greater than that of all other listed models for a given dataset, at a level of significance $\alpha = 0.0011$. The DKL-RBF model (either unpretrained or pretrained with CNN) often outperforms the CNN-1 baselines (unpretrained and pretrained with CNN) on all datasets. All scores are less than the best scores listed in Table 3.3.

Dataset	CNN-2 (scratch)	CNN-2 (pre. CNN)	DKL-RBF (pre. CNN)	DKL-RBF (scratch)
EPR	85.7 ± 0.7	85.6 ± 0.7	86.2 ± 0.5	83.4 ± 0.6
20News-22	67.1 ± 0.7	66.6 ± 0.9	$\mathbf{73.2 \pm 0.9}$	67.8 ± 1.4
IMDB-1	72.4 ± 1.4	72.0 ± 1.5	73.5 ± 1.6	63.6 ± 4.4
20News-100	77.2 ± 0.9	76.8 ± 0.6	$\mathbf{82.8 \pm 0.7}$	77.2 ± 0.9
IMDB-5	79.9 ± 1.0	79.8 ± 0.7	76.8 ± 1.0	$\mathbf{82.1 \pm 1.3}$
IMDB-100	88.4 ± 0.3	88.1 ± 0.4	88.7 ± 0.3	55.7 ± 13.6

Table 2: Mean test F_{macro} scores (as percentages) with standard deviations for each model and dataset. A score is in bold if the mean performance of the model is greater than that of all other listed models for a given dataset, at a level of significance $\alpha = 0.0011$. The F_{macro} scores on the 20-Newsgroups and IMDB datasets are approximately equal to the F_{micro} scores (Table 1) since the test set and test predictions are both nearly class-balanced. The F_{macro} scores on EPR are lower than the F_{micro} scores due to class imbalance but display the same trend across the models.

Dataset	CNN-2 (scratch)	CNN-2 (pre. CNN)	DKL-RBF (pre. CNN)	DKL-RBF (scratch)
EPR	82.0 ± 1.0	82.2 ± 1.0	82.7 ± 0.7	78.7 ± 1.3
20News-22	66.9 ± 0.8	66.2 ± 0.9	$\mathbf{73.1 \pm 0.8}$	67.7 ± 1.4
IMDB-1	72.2 ± 1.5	71.8 ± 1.5	73.4 ± 1.6	62.5 ± 5.3
20News-100	76.8 ± 0.8	76.4 ± 0.7	$\mathbf{82.3 \pm 0.7}$	77.0 ± 0.8
IMDB-5	79.9 ± 1.1	79.8 ± 0.7	76.8 ± 1.0	$\mathbf{82.1 \pm 1.3}$
IMDB-100	88.4 ± 0.3	88.1 ± 0.4	88.7 ± 0.3	55.7 ± 13.6

B Proof of universal kernel approximation theorem

Proof of Thm. 5.1. By Mercer's Theorem [108, Sec. 29], there exist non-negative scalars $\{\lambda_r\}_{r=1}^{\infty}$ and an orthonormal basis of continuous functions $\{e_r : \mathcal{X} \mapsto \mathbb{R}\}_{r=1}^{\infty}$ called "feature maps" such that the target kernel κ admits the representation

$$\kappa(x_1, x_2) = \sum_{r=1}^{\infty} \lambda_r e_r(x_1) e_r(x_2), \tag{1}$$

where the sum on the righthand side converges absolutely and uniformly on the compact set $\mathcal{X}$. Selecting $0 < \delta < \epsilon$, uniform convergence guarantees the existence of an integer R such that for all $x_1, x_2 \in \mathcal{X}$,

$$\left| \sum_{r=1}^{R} \lambda_r e_r(x_1) e_r(x_2) - \kappa(x_1, x_2) \right| < \delta. \tag{2}$$

By continuity of the feature maps e_r and compactness of X, the feature maps are bounded, so there exists a finite positive $B > \max_{r=1,\dots,R} \sqrt{\lambda_r} \sup_{x \in X} |e_r(x)|$. By the classic Universal Approximation Theorem [119], there exists a shallow MLP $\psi : \mathcal{X} \mapsto \mathbb{R}^R$ of width H and parameters $u_{ir} \in \mathbb{R}$ and $\beta_i : \mathcal{X} \mapsto \mathbb{R}$ for $i = 1, \dots, H$ such that for all $x \in \mathcal{X}$,

$$\psi_r(x) = \sum_{i=1}^{H} u_{ir} \sigma(\beta_i(x)) \tag{3}$$

$$\left| \psi_r(x) - \sqrt{\lambda_r} e_r(x) \right| < \gamma < \min\left(B, \frac{\epsilon - \delta}{3NB} \right) \tag{4}$$

$$|\psi_r(x_1)\psi_r(x_2) - \lambda_r e_r(x_1) e_r(x_2)| < \gamma(\gamma + 2B) < 3B\gamma \tag{5}$$

$$\left| \sum_{r=1}^{R} \psi_r(x_1)\psi_r(x_2) - \sum_{r=1}^{R} \lambda_r e_r(x_1) e_r(x_2) \right| < 3NB\gamma < \epsilon - \delta \tag{6}$$

$$\left| \sum_{r=1}^{R} \psi_r(x_1)\psi_r(x_2) - \kappa(x_1, x_2) \right| < (\epsilon - \delta) + \delta = \epsilon. \tag{7}$$

More explicitly, we have the uniform bound

$$\left| \sum_{i,j=1}^{H} \sum_{r=1}^{R} u_{ir} u_{jr} \sigma(\beta_i(x_1)) \sigma(\beta_j(x_2)) - \kappa(x_1, x_2) \right| < \epsilon. \tag{8}$$

Letting $v_{ij} = \sum_{r=1}^{R} u_{ir} u_{jr}$, the bound takes the form claimed in the statement of the theorem. Since this matrix V is of the form UU^T, it is clearly symmetric and positive semi-definite. □

C Derivation of the KL term with optimal prior covariance

Here we derive the optimal prior covariance V_* with respect to the variational free energy in Eq. (5.3) and use it to eliminate the prior covariance V from the loss function entirely.

We note that V appears only in the KL term (Eq. (5.5)) and only through its inverse V^{-1}. Minimizing the KL term with respect to V^{-1}, we obtain Eq. (5.7) as follows:

$$\begin{aligned}\frac{\partial}{\partial V^{-1}}\,\mathrm{KL}(\mu,\Sigma,V)\;|_{V=V_*} &= 0\\ \mu\mu^\top+\Sigma-(V_*^{-1})^{-1} &= 0\\ V_* &= \mu\mu^\top+\Sigma.\end{aligned}$$

Substituting this back into Eq. (5.5) and applying the Matrix Inversion Lemma, we obtain Eq. (5.8) as follows:

$$\begin{aligned}\mathrm{KL}_O(\mu,\Sigma) &= \mathrm{KL}(\mu,\Sigma,V_*)\\ &= \frac{1}{2}\mu^\top(\mu\mu^\top+\Sigma)^{-1}\mu+\frac{1}{2}\,\mathrm{tr}[(\mu\mu^\top+\Sigma)^{-1}\Sigma]-\frac{1}{2}\log\det[(\mu\mu^\top+\Sigma)^{-1}\Sigma]-\frac{H}{2}\\ &= \frac{1}{2}\mu^\top\left(\Sigma^{-1}-\frac{\Sigma^{-1}\mu\mu^\top\Sigma^{-1}}{1+\mu^\top\Sigma^{-1}\mu}\right)\mu\\ &+\frac{1}{2}\,\mathrm{tr}\left[\left(\Sigma^{-1}-\frac{\Sigma^{-1}\mu\mu^\top\Sigma^{-1}}{1+\mu^\top\Sigma^{-1}\mu}\right)\Sigma\right]\\ &+\frac{1}{2}\log\left[\frac{\det(\mu\mu^\top+\Sigma)}{\det\Sigma}\right]-\frac{H}{2}\\ &= \frac{1}{2}\mu^\top\Sigma^{-1}\mu-\frac{1}{2}\frac{\mu^\top\Sigma^{-1}\mu\mu^\top\Sigma^{-1}\mu}{1+\mu^\top\Sigma^{-1}\mu}\\ &+\frac{1}{2}\,\mathrm{tr}\left(I-\frac{\Sigma^{-1}\mu\mu^\top}{1+\mu^\top\Sigma^{-1}\mu}\right)\\ &+\frac{1}{2}\log\left[\frac{\det\Sigma(1+\mu^\top\Sigma^{-1}\mu)}{\det\Sigma}\right]-\frac{H}{2}\\ &= \frac{1}{2}\mu^\top\Sigma^{-1}\mu\left(1-\frac{\mu^\top\Sigma^{-1}\mu}{1+\mu^\top\Sigma^{-1}\mu}\right)\\ &+\frac{H}{2}-\frac{1}{2}\frac{\mathrm{tr}(\Sigma^{-1}\mu\mu^\top)}{1+\mu^\top\Sigma^{-1}\mu}\\ &+\frac{1}{2}\log(1+\mu^\top\Sigma^{-1}\mu)-\frac{H}{2}\\ &= \frac{1}{2}\frac{\mu^\top\Sigma^{-1}\mu}{1+\mu^\top\Sigma^{-1}\mu}\\ &-\frac{1}{2}\frac{\mu^\top\Sigma^{-1}\mu}{1+\mu^\top\Sigma^{-1}\mu}\\ &+\frac{1}{2}\log(1+\mu^\top\Sigma^{-1}\mu)\\ &= \frac{1}{2}\log(1+\mu^\top\Sigma^{-1}\mu).\end{aligned}$$

D Regression results for various numbers of DEKL learners

On the UCI datasets, we evaluate DE and DEKL (both with and without optimal prior covariance) for a varying number of learners to see its effect on test performance (Figs. 1-2). DEKL often maintains superior performance over DE in terms of RMSE as the number of learners is increased, although any trends in the performance gap itself are unclear. We believe an in-depth study on more complex problems and a much greater number of learners is required to see clearer trends, which we leave for future work.

E Diversity of learners on all UCI datasets

Here we present the results of the experiment described in Sec. 5.3.3 on all UCI datasets (Figs. 3-5). Overall, DEKL tends to have more diversity than DE, sometimes by a very large margin; for example, on Kin8nm, the DE learners are almost perfectly correlated while the DEKL learners are almost orthogonal. However, the relationship between diversity and predictive performance is more nebulous. On Boston housing, Concrete, Power plant, and Protein, DEKL has both more diversity and lower RMSE (Table 5.1) than DE. However, on Energy and Yacht, both DE and DEKL have almost perfect correlation among their respective learners, but DEKL achieves lower RMSE; on the other hand, on Kin8nm, Naval propulsion, and Wine, DEKL has greater diversity but not lower RMSE. Understanding when diversity is beneficial for predictive performance is a topic we leave for future work.

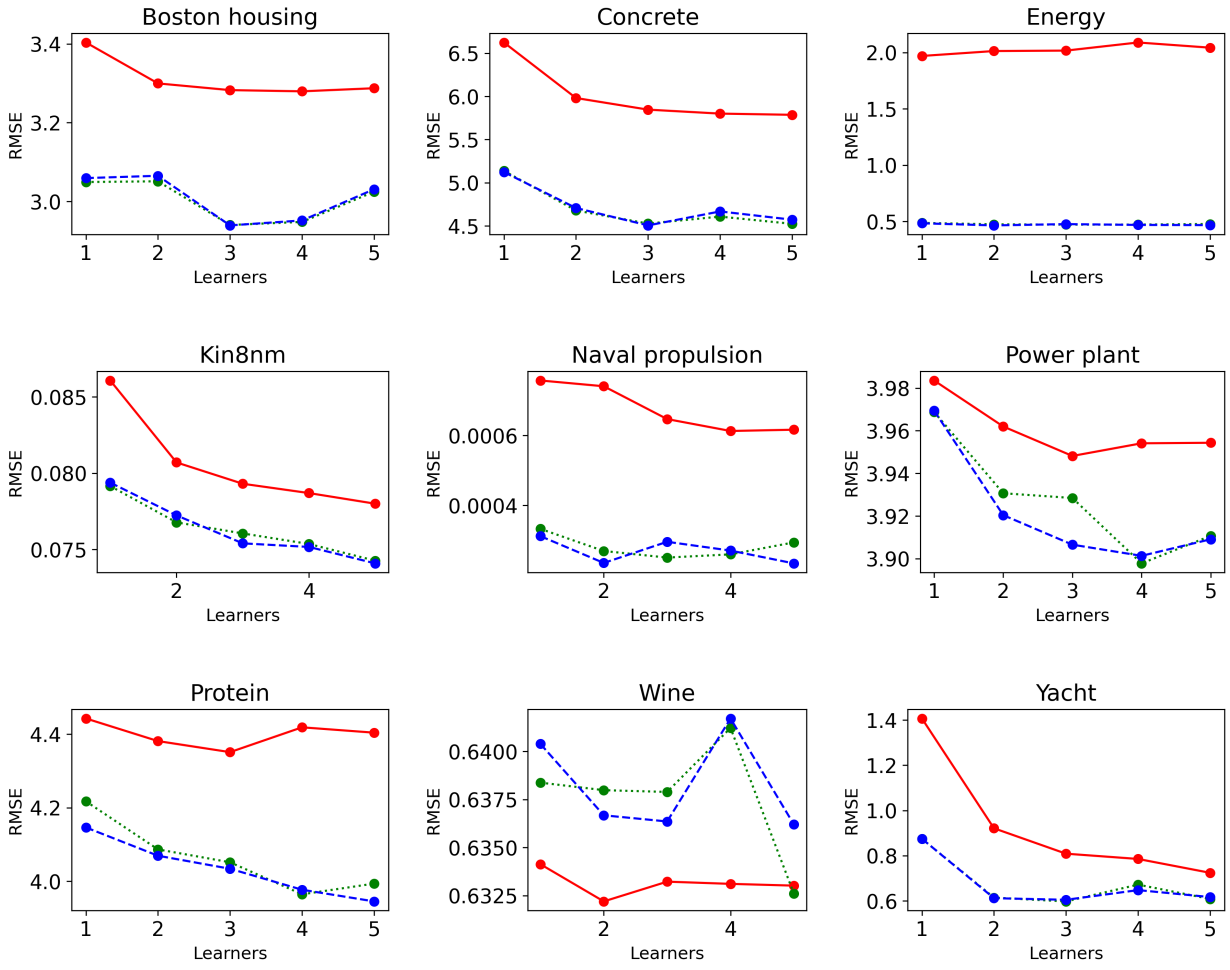


Figure 1: Mean test RMSE (across all train-test splits) of the DE (red, solid), DEKL-I (green, dotted), and DEKL-O (blue, dashed) models on nine UCI datasets as a function of the number of learners.

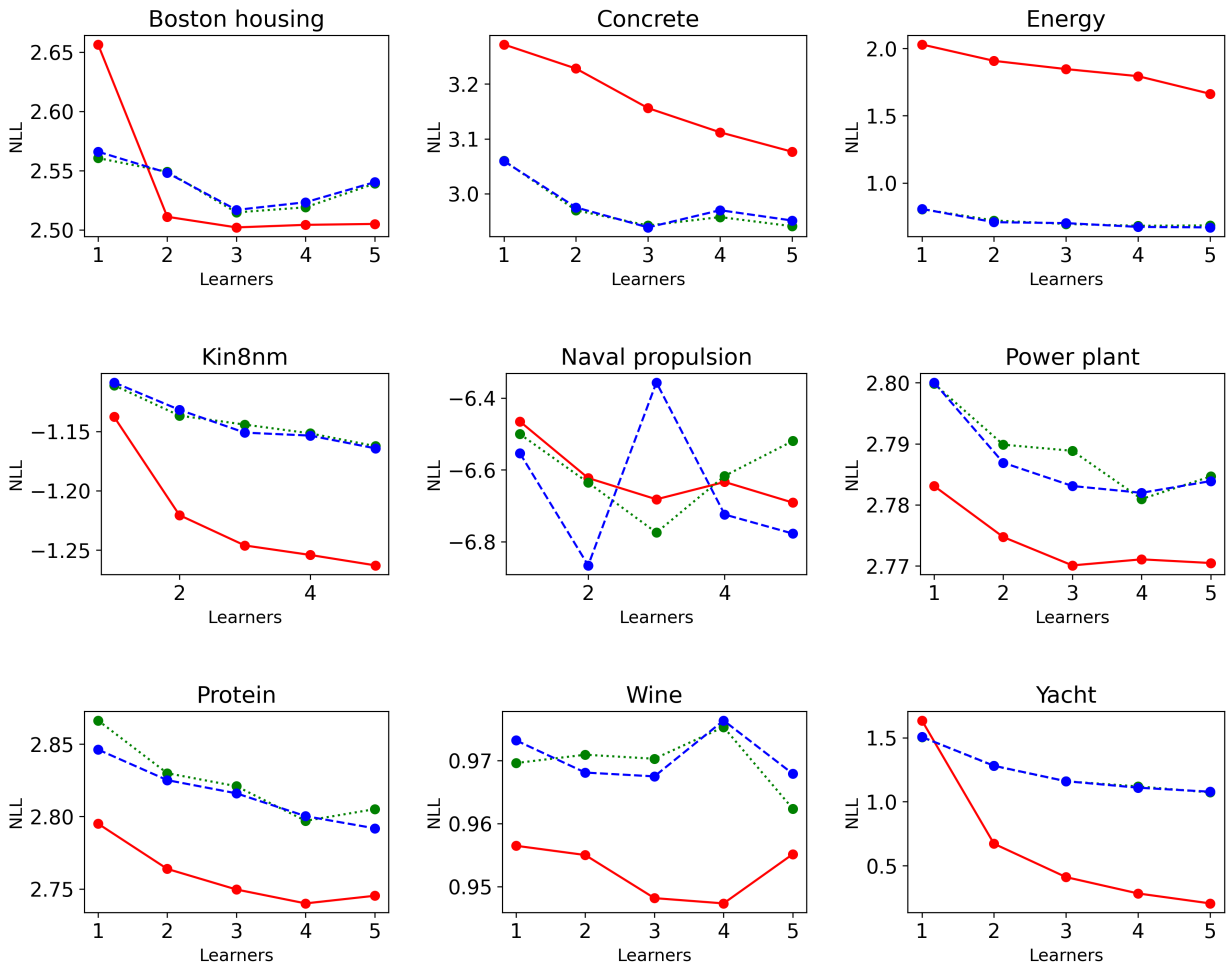


Figure 2: Mean test NLL (across all train-test splits) of the DE (red, solid), DEKL-I (green, dotted), and DEKL-O (blue, dashed) models on nine UCI datasets as a function of the number of learners.

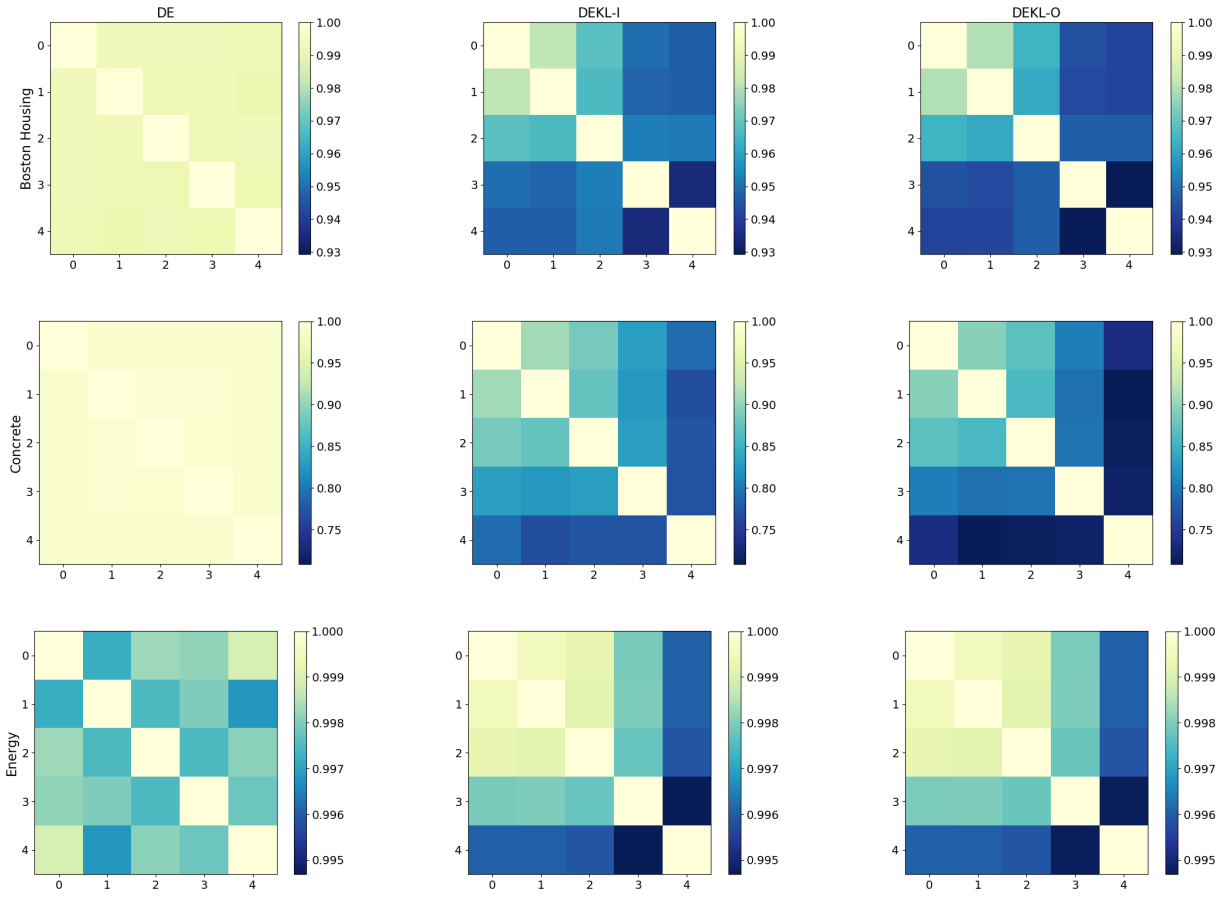


Figure 3: Correlation matrices of learners averaged over all train-test splits on the Boston housing, Concrete, and Energy datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). Suffices -I and -O indicate using KL_I (Eq. (5.9)) and KL_O (Eq. (5.8)), respectively.

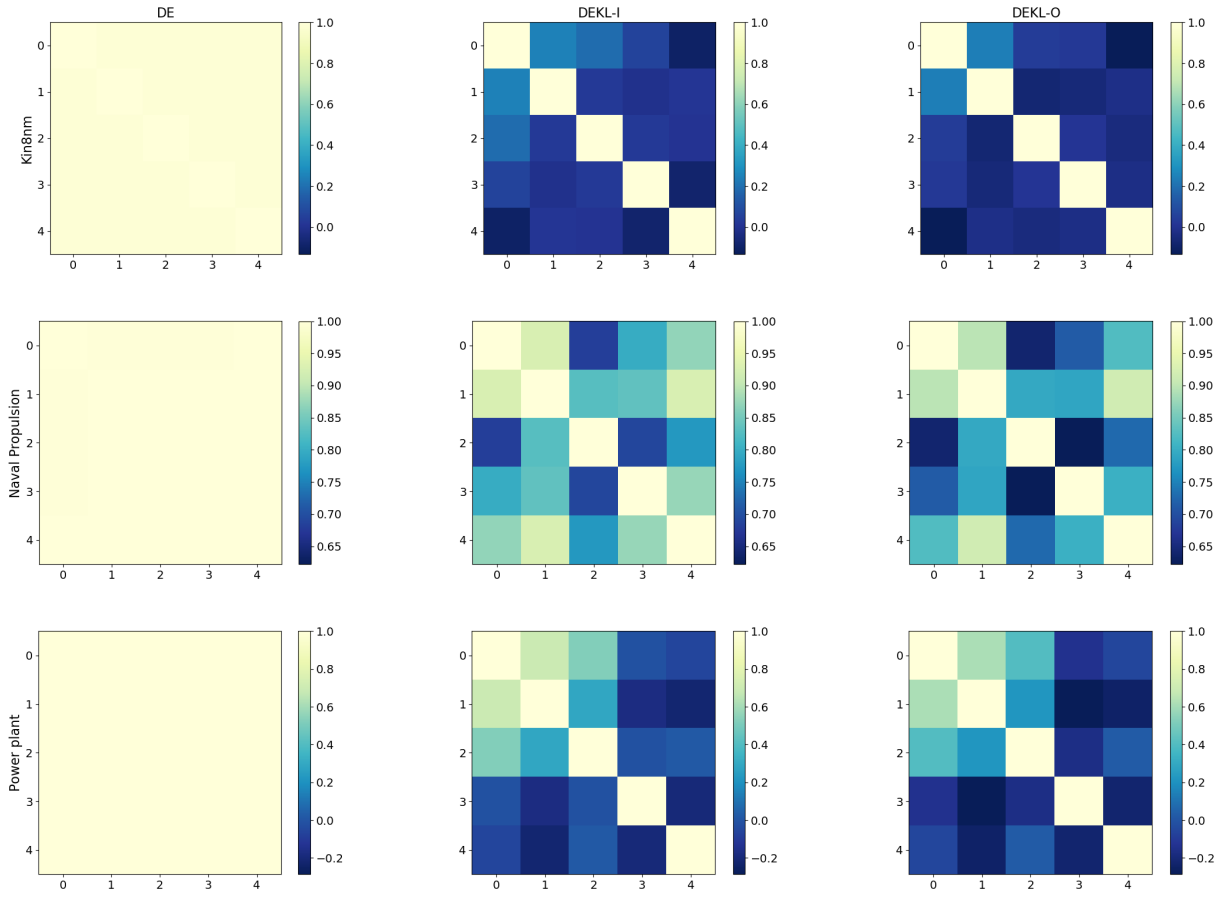


Figure 4: Correlation matrices of learners averaged over all train-test splits on the Kin8nm, Naval propulsion, and Power plant datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). Suffices -I and -O indicate using KL_I (Eq. (5.9)) and KL_O (Eq. (5.8)), respectively.

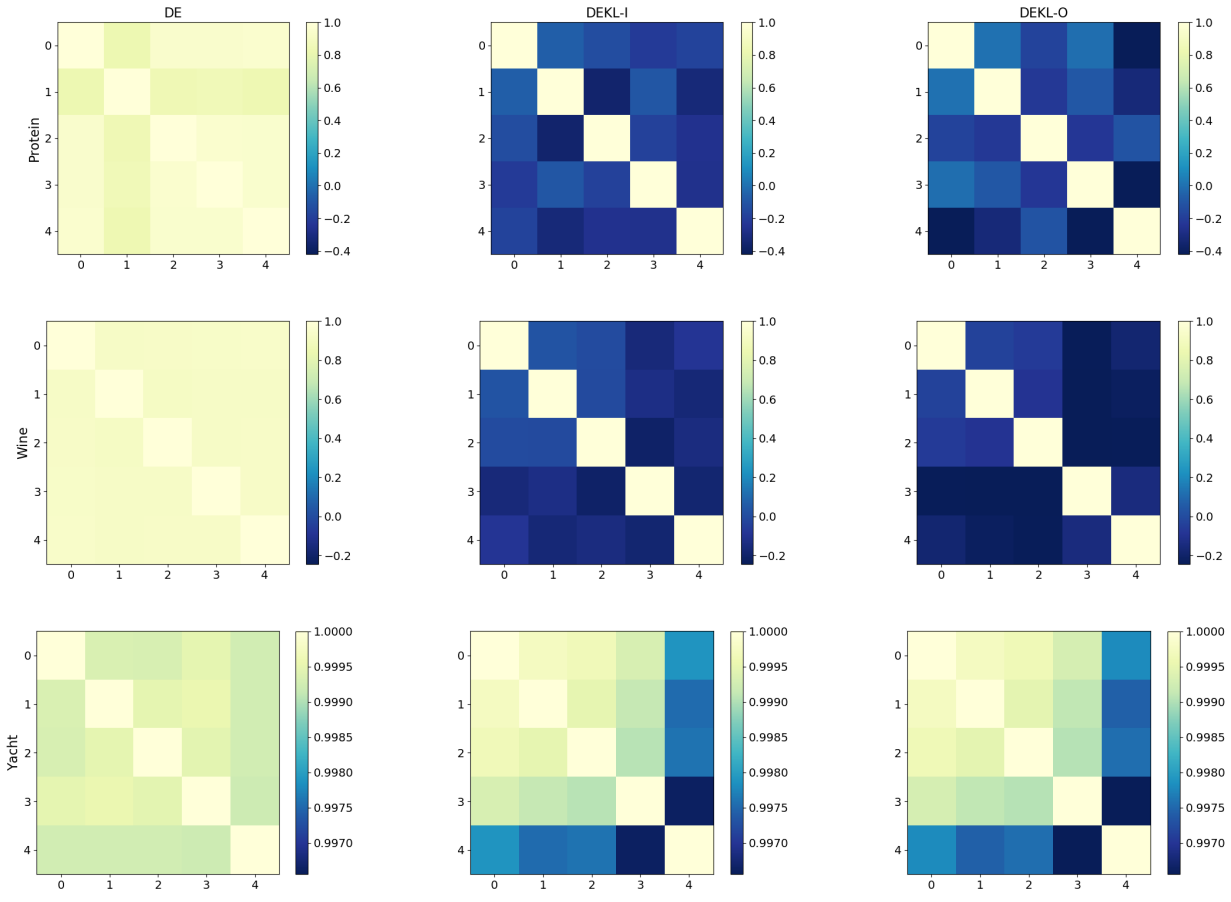


Figure 5: Correlation matrices of learners averaged over all train-test splits on the Protein, Wine, and Yacht datasets using deep ensembles (DE) and deep ensemble kernel learning (DEKL). Suffices -I and -O indicate using KL_I (Eq. (5.9)) and KL_O (Eq. (5.8)), respectively.

F Convergence of stochastic processes

Here we define three notions of convergence that are important for formally stating and proving the bottleneck NNGP limit. The first notion of convergence is the one that Matthews et al., 2018 employ to prove the (no-bottleneck) NNGP limit. They consider a sequence of BNNs mapping $\mathbb{R}^M$ to $\mathbb{R}^L$ restricted to a countable set of inputs $\mathcal{X} \subset \mathbb{R}^M$. Each BNN is then equivalent to a random sequence of L-vectors, i.e. a random variable taking values in $(\mathbb{R}^L)^\infty$. Following Billingsley [16], Matthews et al., 2018 equip the sequence space $(\mathbb{R}^L)^\infty$ with the metric

$$\rho(s,t) = \sum_{k=1}^{\infty} 2^{-k} \min(1, \|s_k - t_k\|),$$

where $s, t \in (\mathbb{R}^L)^\infty$. The metric ρ induces the Euclidean topology on $(\mathbb{R}^L)^\infty$, and we endow $(\mathbb{R}^L)^\infty$ with the associated Borel algebra $\mathcal{A}$, giving us a measurable space of sequences. Demonstrating convergence in $((\mathbb{R}^L)^\infty, \mathcal{A})$ can prove challenging, but Matthews et al., 2018 simplify the task by invoking the following theorem.

Theorem .1 (Billingsley [16])**.** *A sequence of stochastic processes $\{F[n] : \mathcal{X} \times \Omega \mapsto \mathbb{R}^L\}_{n=1}^\infty$ with countable index set $\mathcal{X}$ converges in distribution to a stochastic process $F : \mathcal{X} \times \Omega \mapsto \mathbb{R}^L$ on the measurable space $((\mathbb{R}^L)^\infty, \mathcal{A})$ if and only if every sequence of finite-dimensional marginals $\{(F(x_1)[n], \ldots, F(x_T)[n])\}_{n=1}^\infty$ converges in distribution to the corresponding limiting marginal $(F(x_1), \ldots, F(x_T))$.*

Theorem .1 effectively reduces the task of proving the convergence of a sequence of random sequences to that of a sequence of random vectors. When looking at the convergence of T-dimensional marginal distributions, it is convenient to introduce the following notation.

Definition .2 (Batch stochastic process)**.** *Let $F : \mathbb{R}^M \times \Omega \mapsto \mathbb{R}^L$ be a stochastic process. Then the batch stochastic process of size $T \in \mathbb{N}$ associated with F is the stochastic process $\tilde{F} : (\mathbb{R}^M)^T \times \Omega \mapsto (\mathbb{R}^L)^T$ defined by*

$$\tilde{F}(\{x_t\}_{t=1}^T, \omega) = \{F(x_t, \omega)\}_{t=1}^T.$$

By working with batch stochastic processes, we can think of T inputs as constituting a single input. Thus, to show $\{F[n]\}_{n=1}^\infty \to F$, it is enough to show $\{\tilde{F}(x)[n]\}_{n=1}^\infty \to \tilde{F}(x)$ for each input x for every batch size T.

Our proof of the bottleneck NNGP limit takes the approach of showing that each component of a BNN after the first bottleneck layer converges to an NNGP in the wide limit with some uniformity. We specify the appropriate notion of uniform convergence next.

Definition .3 (Uniform convergence in distribution [137])**.** *A sequence of stochastic processes $\{F[n] : X \times \Omega \mapsto \mathbb{R}^L\}_{n=1}^\infty$ is said to converge in distribution to $\{F : X \times \Omega \mapsto \mathbb{R}^L\}$ uniformly on X if for every continuity set $U \subseteq \mathbb{R}^L$ of F, i.e., a set satisfying*

$$\Pr(F(x) \in \partial U) = 0 \text{ for all } x \in X,$$

we have the limit

$$\lim_{n\to\infty} \Pr(F(x)[n] \in U) = \Pr(F(x) \in U) \text{ uniformly for all } x \in X.$$

We denote this by $F[n] \xrightarrow{UD} F$.

Note that uniform convergence in distribution is distinct from and is not a stronger version of convergence in distribution in $((\mathbb{R}^L)^\infty, \mathcal{A})$ since the former notion concerns only the singly-indexed marginals of a stochastic process while the latter deals with the joint distribution of all elements of a stochastic process. However, the uniform convergence in distribution of the batch stochastic processes $\{\tilde{F}[n]\}_{n=1}^\infty$ is stronger than the convergence in distribution of the original stochastic processes $\{F[n]\}_{n=1}^\infty$. Proving uniform convergence in distribution can be challenging, but fortunately there is another closely related notion of convergence, which we define next.

Definition .4 (Continuous convergence in distribution [137])**.** *Let X be a topological space. A sequence of stochastic processes $\{F[n] : X \times \Omega \mapsto \mathbb{R}^L\}_{n=1}^\infty$ is said to converge in distribution to $\{F : X \times \Omega \mapsto \mathbb{R}^L\}$ continuously on X if for every $x \in X$ and sequence $\{x_n \in X\}_{n=1}^\infty$ converging to x, the sequence of random variables $\{F(x_n)[n]\}_{n=1}^\infty$ converges in distribution to $F(x)$. We denote this by $F[n] \xrightarrow{CD} F$.*

Uniform and continuous convergence in distribution are related through the following proposition.

Proposition .5 (Saikkonen [128])**.** *Let X be a topological space. Let $\{F[n] : X \times \Omega \mapsto \mathbb{R}^L\}_{n=1}^\infty$ be a sequence of stochastic processes and $\{F : X \times \Omega \mapsto \mathbb{R}^L\}$ a stochastic process. Then the following are equivalent statements:*

(a) $F[n] \xrightarrow{CD} F$.

(b) $F[n] \xrightarrow{UD} F$ *on every compact subset of* X *and* $x \mapsto \Pr(F(x) \in U)$ *is a continuous function for every continuity set* U *of* F.

We should mention that Sweeting [137] and Saikkonen [128] do not state Defs. .3-.4 and Prop. .5 directly but instead define and work with the equivalent notions of uniform and continuous weak convergence of probability measures. The proof of Prop. .5 is a simple application of the equivalence of uniform and continuous convergence of real-valued functions on compact sets.

G The bottleneck NNGP theorem

Here we list the proof of our main theorem (Thm. 6.4) along with all supporting lemmas. We give a sketch of the proof next, highlighting the role of each lemma and its position in the general proof strategy.

The first step is to apply Thm. .1 so that it is sufficient to prove convergence of BNNs to a bottleneck NNGP restricted to an arbitrary finite set of inputs. Since each component $F^{(d)}[n]$ in Thm. 6.4 is being evaluated at T inputs, then it is convenient to utilize the concept of a batch stochastic process of size T (Def. .2). By working with batch BNNs, we can think of T inputs as constituting a single "batch" input. This reduces our task to proving the convergence of batch BNNs to a batch bottleneck NNGP given a single arbitrary input.

The next step is to find sufficient conditions under which the distributional limit of an element-wise composition of a sequence of stochastic processes with a sequence of random

variables equals the composition of the limiting stochastic process with the limiting random variable. This trick can then be iterated via induction to prove that the limit of compositions is the composition of limits for stochastic processes. Lemma .7 provides such sufficient conditions, which include a notion of uniform convergence in distribution (Def. .3).

Proving Thm. 6.4 now comes down to verifying the conditions of Lemma .7. Condition (a) is given to us by Thm. 6.3 for single-bottleneck networks and holds by induction for multi-bottleneck networks. Condition (b) is also immediate in the single-bottleneck case, although it is less obvious for multi-bottleneck architectures. We verify condition (b) directly in the proof of Thm. 6.4 but take aid from Lemma .8. Lemma .9 establishes condition (c), which amounts to showing that the NNGP kernel is a continuous function. Condition (d) is the trickiest to verify; it states that the outer sequence of stochastic processes (that is composed with an inner sequence of random variables) must converge in distribution uniformly (Def. .3) on compact sets, meaning that the rate of convergence in distribution should be independent of the input to the stochastic processes. Condition (d) is verified with the help of Lemma .10.

Lemma .10 is a direct generalization of Lemma 12 in Matthews et al., 2018; the latter states that given a fixed finite batch of inputs, BNNs with no bottlenecks converge in distribution to an NNGP in the wide limit. In Lemma .10, we strengthen the mode of convergence to continuous convergence in distribution (Def. .4). More specifically, Lemma .10 states that a BNN converges in distribution to an NNGP even if we replace the fixed batch of inputs with a convergent sequence of input batches. Continuous convergence in distribution is in fact equivalent to uniform convergence in distribution on compact sets (Prop. .5), thus granting condition (d).

The proof of Lemma .10 runs in parallel to the proof of Lemma 12 in Matthews et al., 2018. It depends on several lemmas (Appendices G.3-G.4) that are all simple extensions of (or help to extend) the lemmas in Matthews et al., 2018; at each step, we simply replace the fixed batch of inputs in Matthews et al., 2018 with a convergent sequence of input batches and verify that convergence in distribution still holds. Only a few key modifications are made to the lemmas establishing uniform integrability (Appendix G.4).

G.1 Notation

We start with some notation. Let $\{F[n] : \mathbb{R}^M \times \Omega \mapsto \mathbb{R}^L\}_{n=1}^{\infty}$ be a sequence of BNNs each with D hidden layers of widths $H_\mu[n]$, $\mu \in \{1, \ldots, D\}$, and nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$ that satisfies the linear envelope condition. Let $F : \mathbb{R}^M \times \Omega \mapsto \mathbb{R}^L$ be the limiting NNGP of the sequence of BNNs. Let $f_i^{(\mu)}(x)[n]$ (resp. $f_i^{(\mu)}(x)$) and $g_i^{(\mu)}(x)[n]$ (resp. $g_i^{(\mu)}(x)$) be the preactivation and activation of the i-th neuron in the μ-th hidden layer of $F[n]$ (resp. F).

For each $n \in \mathbb{N}$, let $X[n] = \{x_t[n] \in \mathbb{R}^M\}_{t=1}^T$ be a batch of T inputs, and suppose the sequence of batches $\{X[n]\}_{n=1}^{\infty}$ converges to some finite $X = \{x_t\}_{t=1}^T$. Let $\alpha \in \mathbb{R}^{T \times |\mathbb{N}|}$ be a countably infinite block vector whose blocks are indexed by $\mathbb{N}$ and where each block has T elements. Let α have finite support $\{1, \ldots, T\} \times I$, where I is a finite subset of $\mathbb{N}$; i.e., only finitely many blocks indexed by I are permitted to have non-zero elements. Let α_{ti} denote the t-th element in the i-th block. For each $\mu \in \{1, \ldots, D+1\}$, define the preactivation

projections of a BNN and its limiting NNGP as

$$f^{(\mu)}(X[n], \alpha)[n] = \sum_{t=1}^{T} \sum_{i \in I} \alpha_{ti} f_i^{(\mu)}(x_t[n])[n],$$
$$f^{(\mu)}(X, \alpha) = \sum_{t=1}^{T} \sum_{i \in I} \alpha_{ti} f_i^{(\mu)}(x_t).$$

Let $k^{(\mu)} : \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}$ be the NNGP kernel of $f_i^{(\mu)}$. The kernel $k^{(\mu)}$ relates to the block kernel $K^{(\mu)}$ (Eqs. (6.6)-(6.7)) through the equation

$$K^{(\mu)}(X, X) = \begin{cases} k^{(\mu)}(X, X) \otimes \mathbf{I}_\infty & \text{for } \mu \in \{1, \ldots, D\} \\ k^{(\mu)}(X, X) \otimes \mathbf{I}_L & \text{for } \mu = D + 1, \end{cases}$$

where $\mathbf{I}_L$ is the $L \times L$ identity matrix, $\mathbf{I}_\infty$ is the countably infinite identity matrix, and $\otimes$ denotes the Kronecker product. We also define

$$L_{ij}^{(\mu)}(x_1, x_2) = \mathrm{Cov}[g_i^{(\mu)}(x_1), g_j^{(\mu)}(x_2)] = \delta_{ij} \mathrm{Cov}[g_1^{(\mu)}(x_1), g_1^{(\mu)}(x_2)],$$

which satisfies the relation

$$K_{ij}^{(\mu+1)}(x_1, x_2) = v_b \delta_{ij} + v_w L_{ij}^{(\mu)}(x_1, x_2).$$

We let $L^{(\mu)}(X, X)$ denote a block matrix where $L_{ij}^{(\mu)}(X, X)$ is the (i, j)-th block. The block matrices $K^{(\mu)}(X, X)$ and $L^{(\mu)}(X, X)$ for $\mu \in \{1, \ldots, D\}$ have infinitely many blocks since the NNGP has infinitely wide hidden layers. However, given a block vector $\alpha \in \mathbb{R}^{T \times |\mathbb{N}|}$ of finite support $\{1, \ldots, T\} \times I$, the quadratic forms $\alpha^\top K^{(\mu)}(X, X)\alpha$ and $\alpha^\top L^{(\mu)}(X, X)\alpha$ are still finite sums:

$$\begin{aligned} \alpha^\top K^{(\mu)}(X, X)\alpha &= \sum_{t,u=1}^{T} \sum_{i,j \in I} \alpha_{ti} \alpha_{uj} K^{(\mu)}(X, X)_{T(i-1)+t, T(j-1)+u} \\ &= \mathrm{V}[\sum_{t=1}^{T} \sum_{i \in I} \alpha_{ti} f_i^{(\mu)}(x_t)] \\ &= \mathrm{V}[f^{(\mu)}(X, \alpha)], \end{aligned}$$

We define a quadratic form for $L^{(\mu)}(X, X)$ similarly.

Next we define the quantities that are at the heart of the proof of Lemma .10. This definition is similar to Definition 7 in Matthews et al., 2018. We discuss the purpose of this definition in more detail in Appendix G.3 in the context of the Central Limit Theorem.

Definition .6 (Projections and summands)**.** *For each $\mu \in \{2, \ldots, D+1\}$ and for each $n \in \mathbb{N}$ and $j \in \{1, \ldots, n\}$, define the summands*

$$\gamma_j^{(\mu)}(X[n], \alpha)[n] = \sqrt{H_{\mu-1}[n]} \sum_{t=1}^{T} \sum_{i \in I} \alpha_{ti} w_{ij}^{(\mu)} g_j^{(\mu-1)}(x_t[n])[n], \tag{9}$$

and the projections

$$\begin{aligned} S^{(\mu)}(X[n],\alpha)[n] &= \sum_{t=1}^{T}\sum_{i\in I}\alpha_{ti}\left(f_i^{(\mu)}(x_t[n])[n]-b_i^{(\mu)}\right) \\ &= \frac{1}{\sqrt{H_{\mu-1}[n]}}\sum_{j=1}^{H_{\mu-1}[n]}\gamma_j^{(\mu)}[n]. \end{aligned} \tag{10}$$

Finally, for $\mu \in \{1,\dots,D+1\}$, define the variances

$$\sigma^2_{(\mu)}(X[n],\alpha)[n] = \mathrm{V}[\gamma_j^{(\mu)}(X[n],\alpha)[n]] \tag{11}$$

$$\begin{aligned} \sigma^2_{(\mu)}(X,\alpha) &= v_w\alpha^\top L^{(\mu-1)}(X,X)\alpha \\ &= \alpha^\top\left[K^{(\mu)}(X,X)-v_b\mathbf{I}\otimes\mathbf{1}_{T\times T}\right]\alpha. \end{aligned} \tag{12}$$

G.2 Main lemmas and theorem

This section contains the proof of the main theorem (Thm. 6.4) as well as all original lemmas supporting it. See the proof sketch in Sec. 6.3.1 for an overview and guide to the logical flow of the lemmas. We start with a lemma that gives sufficient conditions under which the distributional limit of a sequence of compositions of stochastic processes and random variable indices equals the composition of limits.

Lemma .7 (Limit of stochastic process compositions)**.** *Let $\{X[n]\}_{n=1}^{\infty}$ be a sequence of random vectors and X a random vector of dimension B. Let $\{F[n]:\mathbb{R}^B\times\Omega\mapsto\mathbb{R}^L\}_{n=1}^{\infty}$ be a sequence of stochastic processes and $F:\mathbb{R}^B\times\Omega\mapsto\mathbb{R}^L$ a stochastic process with $F(x)\sim\mathcal{N}(0,\Sigma(x))$. If*

(a) $X[n]$ converges in distribution to X, denoted $X[n]\xrightarrow{D}X$,

(b) $\{\mathrm{E}\{|X[n]|^2\}\}_{n=1}^{\infty}$ is eventually bounded,

(c) $\Sigma:\mathbb{R}^B\mapsto\mathbb{R}^{L\times L}$ is a continuous function, and

(d) $F[n]\xrightarrow{UD}F$ on every compact ball in $\mathbb{R}^B$ centered at 0,

then the sequence of random variables $F(X[n])[n]\xrightarrow{D}F(X)$.

Proof. We first prove the claim for the case that $F[n]$ and F are real-valued stochastic processes ($L=1$) and $\Sigma(x)>0$ for all $x\in\mathbb{R}^B$. For this case, we will use the notation $\sigma^2(x)$ in place of $\Sigma(x)$ to emphasize that $\Sigma(x)$ is a scalar.

Let c be a continuity point of $F(X)$. We want to show that

$$\lim_{n\to\infty}\Pr(F(X[n])[n]<c)=\Pr(F(X)<c).$$

Let $\varepsilon>0$. We have

$$\begin{aligned} &|\Pr(F(X[n])[n]<c)-\Pr(F(X)<c)| \\ &=|\Pr(F(X[n])[n]<c)-\Pr(F(X[n])<c)+\Pr(F(X[n])<c)-\Pr(F(X)<c)| \\ &\le|\Pr(F(X[n])[n]<c)-\Pr(F(X[n])<c)|+|\Pr(F(X[n])<c)-\Pr(F(X)<c)|. \end{aligned} \tag{13}$$

We will show that both terms on the right-hand side of Inequality (13) tend to 0.

We start with the second term. Let $\mu[n]$ and μ be the probability distributions associated with $X[n]$ and X, respectively. Then the second term becomes

$$|\Pr(F(X[n]) < c\} - \Pr(F(X) < c\}| = \left| \int_{\mathbb{R}^B} \Pr(F(x) < c)\, \mathrm{d}\mu(x)[n] - \int_{\mathbb{R}^B} \Pr(F(x) < c)\, \mathrm{d}\mu(x) \right|.$$

Since $F(x) \sim \mathcal{N}(0, \sigma^2(x))$ with $\sigma^2(x) > 0$, then we have

$$\Pr(F(x) < c) = \Phi\left(\frac{c}{\sigma(x)}\right),$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution. Since σ is continuous, then the map $x \mapsto \Pr(F(x) < c)$ is immediately seen to be continuous as well. Moreover, $\Pr(F(x) < c)$ is clearly bounded. Since $X[n] \xrightarrow{D} X$, then $\{\mu[n]\}_{n=1}^{\infty}$ converges weakly to μ, so that

$$\lim_{n\to\infty} \int_{\mathbb{R}^B} \Pr(F(x) < c)\, \mathrm{d}\mu(x)[n] = \int_{\mathbb{R}^B} \Pr(F(x) < c)\, \mathrm{d}\mu(x).$$

Therefore, there exists an integer N_2 such that

$$|\Pr(F(X[n]) < c) - \Pr(F(X) < c)| < \frac{\varepsilon}{2} \text{ for all } n > N_2. \tag{14}$$

We next bound the first term on the right-hand side of Inequality (13). We have

$$\begin{aligned} &|\Pr(F(X[n])[n] < c) - \Pr(F(X[n]) < c)| \\ &= \left| \int_{\mathbb{R}^B} \Pr(F(x)[n] < c)\, \mathrm{d}\mu(x)[n] - \int_{\mathbb{R}^B} \Pr(F(x) < c)\, \mathrm{d}\mu(x)[n] \right| \\ &\leq \int_{\mathbb{R}^B} |\Pr(F(x)[n] < c) - \Pr(F(x) < c)|\, \mathrm{d}\mu(x)[n]. \end{aligned} \tag{15}$$

We will bound the integrand. Since $\{\mathrm{E}\{|X[n]|^2\}\}_{n=1}^{\infty}$ is eventually bounded, then there exists $V > 0$ and an integer N_V such that

$$\mathrm{E}\{|X[n]|^2\} < V \text{ for all } n > N_V.$$

Define

$$R_\varepsilon = \sqrt{\max\left(0, \frac{2}{\varepsilon}(1+V) - 1\right)}.$$

R_ε is defined such that $\|x\| > R_\varepsilon$ implies

$$\frac{\varepsilon}{2} \cdot \frac{1 + \|x\|^2}{1+V} > 1.$$

We therefore have

$$|\Pr(F[n](x) < c) - \Pr(F(x) < c)| \leq 1 < \frac{\varepsilon}{2} \cdot \frac{1 + \|x\|^2}{1+V} \text{ for all } x \mid \|x\| > R_\varepsilon. \tag{16}$$

Since $F(x)$ follows a normal distribution, then c is trivially a continuity point of $F(x)$ for every $x \in \mathbb{R}^B$. Since $\{F[n]\}_{n=1}^{\infty}$ converges in distribution to F uniformly on every zero-centered compact ball, then there exists an integer $N_1 > N_V$ such that

$$|\Pr(F(x)[n] < c) - \Pr(F(x) < c)| < \frac{\varepsilon}{2} \cdot \frac{1}{1+V} \text{ for all } n > N_1 \text{ and } \|x\| \leq R_\varepsilon.$$

Since $\|x\|^2 \geq 0$, then we have the weaker bound

$$|\Pr(F(x)[n] < c) - \Pr(F(x) < c)| < \frac{\varepsilon}{2} \cdot \frac{1+\|x\|^2}{1+V} \text{ for all } n > N_1 \text{ and } \|x\| \leq R_\varepsilon.$$

Combining this with Eq. (16) gives

$$|\Pr(F(x)[n] < c) - \Pr(F(x) < c)| < \frac{\varepsilon}{2} \cdot \frac{1+\|x\|^2}{1+V} \text{ for all } n > N_1 \text{ and } x \in \mathbb{R}^B.$$

Using this bound in Inequality (15), we get

$$\begin{aligned} |\Pr(F(X[n])[n] < c) - \Pr(F(X[n]) < c)| &\leq \int_{\mathbb{R}^B} \frac{\varepsilon}{2} \cdot \frac{1+\|x\|^2}{1+V} \, \mathrm{d}\mu_n(x) \\ &= \frac{\varepsilon}{2} \cdot \frac{1+\mathrm{E}\{|X[n]|^2\}}{1+V} \\ &\leq \frac{\varepsilon}{2} \cdot \frac{1+V}{1+V} \\ &= \frac{\varepsilon}{2} \text{ for all } n > N_1. \end{aligned} \tag{17}$$

Let $N = \max(N_1, N_2)$. Combining Inequalities (13), (14), and (17), we obtain the bound

$$|\Pr(F(X[n])[n] < c) - \Pr(F(X) < c)| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \text{ for all } n > N,$$

implying $F(X[n])[n] \xrightarrow{D} F(X)$.

Now consider the more general case where the processes $F[n]$ and F take values in $\mathbb{R}^L$ for $L \geq 1$ and where the kernel k of F is not necessarily strictly positive definite. Consider any $\alpha \in \mathbb{R}^L$, and define the processes

$$\begin{aligned} \hat{F}(x)[n] &= Z + \alpha^\top F(x)[n], \\ \hat{F}(x) &= Z + \alpha^\top F(x), \end{aligned}$$

where $Z \sim \mathcal{N}(0,1)$ is independent of $F[n]$ and F. $\hat{F}[n]$ and $\hat{F}$ are real-valued stochastic processes and $\hat{F}(x)$ is normally distributed with variance

$$\hat{\sigma}^2(x) = 1 + \alpha^\top \Sigma(x) \alpha > 0.$$

By the case already proven above, $\hat{F}(X[n])[n] \xrightarrow{D} \hat{F}(X)$. Since the addition of an independent normally distributed random variable Z preserves convergence in distribution, then $\alpha^\top F(X[n])[n] \xrightarrow{D} \alpha F(x)$. Since this holds for any vector α, then by the Cramér-Wold Device [15], we obtain the conclusion $F(X[n])[n] \xrightarrow{D} F(X)$. □

Lemma .7 can be applied inductively to show that a sequence of compositions of stochastic processes converges in distribution to the composition of limit processes. The next lemma verifies condition (b) of Lemma .7.

Lemma .8 (Uniformly bounded neural network variances). *Let $\{F[n]\}_{n=1}^{\infty}$ be a sequence of BNNs mapping $\mathbb{R}^M$ to $\mathbb{R}^L$ with D hidden layers of widths $H_\mu[n]$, $\mu \in \{1, \ldots, D\}$, and nonlinearity ϕ^μ on the μ-th hidden layer satisfying the linear envelope condition. Then for every $x \in \mathbb{R}^M$, the sequence of second moments $\mathrm{E}[\|F(x)[n]\|^2]\}_{n=1}^{\infty}$ is uniformly bounded by $A + B\|x\|^2$ for some constants $A, B > 0$.*

Proof. Let $x \in \mathbb{R}^M$. The claimed uniform bound on $\mathrm{E}[\|F(x)[n]\|^2]$ holds if we can uniformly bound $\{\mathrm{E}[\|F_i(x)[n]\|^2]\}_{n=1}^{\infty}$ for each i. Since $F_i(x)[n] = f_i^{(D+1)}(x)[n]$, then we need to establish $\mathrm{E}[f_i^{(D+1)}(x)[n]^2] \leq A + B\|x\|^2$ for sufficiently large n. We proceed by induction on μ. In the case $\mu = 1$, we have

$$\mathrm{E}[f_i^{(1)}(x)[n]] = v_b + v_w\|x\|^2.$$

Taking $A = v_b$ and $B = v_w$, this is clearly bounded by $A + B\|x\|^2$ independently of n. By exchangeability, this same bound holds for all i.

Now suppose for some μ that the claimed uniform bound holds. We then need to establish the bound $\mathrm{E}[f_i^{(\mu+1)}(x)[n]] \leq A + B\|x\|^2$ for some $A, B > 0$. We have

$$\begin{aligned}
\mathrm{E}[f_i^{(\mu+1)}(x)[n]^2] &= v_b + \frac{v_w}{H_\mu[n]} \sum_{j=1}^{H_\mu[n]} \mathrm{E}[g_j^{(\mu)}(x)[n]^2] \\
&= v_b + v_w \,\mathrm{E}[g_i^{(\mu)}(x)[n]^2] \\
&= v_b + v_w \int_{-\infty}^{\infty} \phi^{(\mu)}(z)^2 \,\mathrm{d}\mu_n(z),
\end{aligned}$$

where μ_n is the probability distribution of $f_i^{(\mu)}(x)[n]$. By the linear envelope condition,

$$\begin{aligned}
\mathrm{E}[f_i^{(\mu+1)}(x)[n]^2] &\leq v_b + v_w \int_{-\infty}^{\infty} (C + M|z|)^2 \,\mathrm{d}\mu_n(z) \\
&\leq v_b + v_w \int_{-\infty}^{\infty} 2(C^2 + M^2|z|^2) \,\mathrm{d}\mu_n(z) \\
&= v_b + 2v_w(C^2 + M^2 \,\mathrm{E}[f_i^{(\mu)}(x)[n]^2]) \\
&\leq v_b + 2v_w(C^2 + M^2(A + B\|x\|^2)),
\end{aligned}$$

which is clearly bounded by an expression of the form $A' + B'\|x\|^2$ independently of n for some $A', B' > 0$. The claim then follows by induction. □

The next lemma verifies condition (c) of Lemma .7, which amounts to showing that the NNGP kernel is continuous.

Lemma .9 (Continuity of batch NNGP kernel). *Let $F : \mathbb{R}^M \times \Omega \mapsto \mathbb{R}^L$ be an NNGP with D hidden layers and nonlinearity ϕ that satisfies the linear envelope condition. Then the associated batch NNGP $\tilde{F} : (\mathbb{R}^M)^T \times \Omega \mapsto (\mathbb{R}^L)^T$ of size T has marginal $\tilde{F}(X) \sim \mathcal{N}(0, \Sigma(X))$, where the batch NNGP kernel $\Sigma : (\mathbb{R}^M)^T \mapsto \mathbb{R}^{LT \times LT}$ given by $\Sigma(X) = K(X, X)$ is a continuous function.*

Proof. All we need to show is that Σ is a continuous function. Since $\Sigma(X) = K(X, X) = k(X, X) \otimes I_L$, then it is sufficient to show that the NNGP kernel $k : \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}$ is continuous. We do so inductively by showing that $k^{(\mu)}$ is continuous for $\mu \in \{1, \ldots, D+1\}$, where $k = k^{(D+1)}$.

For $\mu = 1$, $k^{(1)}(x, x') = v_b + v_w x \cdot x'$ is clearly continuous. Now suppose for some $\mu \in \{1, \ldots, D\}$, that $k^{(\mu)}$ is continuous. We then need to show that $k^{(\mu+1)}$ is continuous. Let $\{X_n = (x_n, x_n') \in \mathbb{R}^M \times \mathbb{R}^M\}_{n=1}^{\infty}$ be a convergent sequence of pairs of inputs such that $X_n \to X = (x, x')$. Since $k^{(\mu)}$ is continuous, then $k^{(\mu)}(X_n, X_n) \to k^{(\mu)}(X, X)$. Let $s_i(X_n)$ (resp. $s_i(X)$) denote the i-th column of the symmetric positive semidefinite square root $S(X_n)$ (resp. $S(X)$) of $k^{(\mu)}(X_n, X_n)$ (resp. $k^{(\mu)}(X, X)$). Then by the work of Cho and Saul [28], the kernel recursion in Eq. (6.7) can be expressed as

$$k^{(\mu+1)}(x_n, x_n') = v_b + v_w \frac{1}{2\pi} \int_{\mathbb{R}^2} \phi(w^\top s_1(X_n)) \phi(w^\top s_2(X_n))\, e^{-\frac{\|w\|^2}{2}}\, \mathrm{d}w,$$

and $k^{(\mu+1)}(x, x')$ is given similarly. To ensure continuity, we will show that $k^{(\mu+1)}(x_n, x_n') \to k^{(\mu+1)}(x, x')$. We do so by verifying the conditions of the Dominated Convergence Theorem. First, by the linear envelope condition, there exist positive constants C and M such that

$$\begin{aligned}
\phi(w^\top s_1(X_n)) \phi(w^\top s_2(X_n)) &\leq [C + M w^\top s_1(X_n)][C + M w^\top s_2(X_n)] \\
&= C^2 + CM \sum_{i=1}^{2} w^\top s_i(X_n) + M^2 w^\top s_1(X_n) w^\top s_2(X_n) \\
&\leq C^2 + CM \sum_{i=1}^{2} w^\top s_i(X_n) + \frac{M^2}{2} \sum_{i=1}^{2} (w^\top s_i(X_n))^2 \\
&= \frac{1}{2} \left[(C + M w^\top s_1(X_n))^2 + (C + M w^\top s_2(X_n))^2 \right] \\
&\leq [C^2 + M^2 (w^\top s_1(X_n))^2] + [C^2 + M^2 (w^\top s_2(X_n))^2] \\
&= 2C^2 + M^2 \|S^\top(X_n) w\|^2 \\
&\leq 2C^2 + M^2 \|S^\top(X_n)\|_2^2 \|w\|^2 \\
&\leq 2C^2 + M^2 \|S^\top(X_n)\|_F^2 \|w\|^2 \\
&= 2C^2 + M^2 \|S(X_n)\|_F^2 \|w\|^2,
\end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Since the matrix square root operation is continuous under the Frobenius norm, then $\|S(X_n)\|_F$ is bounded by some $B > 0$. We therefore have

$$\phi(w^\top s_1(X_n)) \phi(w^\top s_2(X_n)) \leq 2C^2 + M^2 B^2 \|w\|^2,$$

hence

$$\frac{1}{2\pi} \phi(w^\top s_1(X_n)) \phi(w^\top s_2(X_n)) e^{-\frac{\|w\|^2}{2}} \leq \frac{1}{2\pi} (2C^2 + M^2 B^2 \|w\|^2) e^{-\frac{\|w\|^2}{2}},$$

where the bound on the right-hand side is clearly integrable over $w \in \mathbb{R}^2$. Moreover, since the matrix square root operation is continuous and ϕ is continuous, then

$$\frac{1}{2\pi} \phi(w^\top s_1(X_n)) \phi(w^\top s_2(X_n)) e^{-\frac{\|w\|^2}{2}} \to \frac{1}{2\pi} \phi(w^\top s_1(X)) \phi(w^\top s_2(X)) e^{-\frac{\|w\|^2}{2}} \text{ pointwise in } w. \tag{18}$$

Therefore, by the Dominated Convergence Theorem, $k^{(\mu+1)}(x_n, x_n') \to k^{(\mu+1)}(x, x')$ so that $k^{(\mu+1)}$ is continuous. The continuity of the kernel k then follows by induction. □

The next lemma will help to verify condition (d) of Lemma .7. It is a generalization of Lemma 12 in Matthews et al., 2018 and depends on several additional lemmas (Appendices G.3-G.4) similar to those in Matthews et al., 2018.

Lemma .10 (Continuous convergence in distribution of batch BNNs)**.** *Consider a sequence $\{F[n]\}_{n=1}^{\infty}$ of BNNs mapping $\mathbb{R}^M$ to $\mathbb{R}^L$ with D hidden layers of widths $H_\mu[n]$, $\mu \in \{1, \ldots, D\}$, and nonlinearity ϕ that satisfies the linear envelope condition. Let F be the NNGP limit of the BNNs as given by Thm. 6.3. Then for any $T \in \mathbb{N}$, the corresponding sequence of batch BNNs $\{\tilde{F}[n]\}_{n=1}^{\infty}$ converges in distribution to the batch NNGP $\tilde{F}$ continuously.*

Proof. For each $n \in \mathbb{N}$, let $X[n] = \{x_t[n]\}_{t=1}^{T} \in (\mathbb{R}^M)^T$ be a batch of inputs such that the sequence of batches $\{X[n]\}_{n=1}^{\infty}$ converges to some finite $X \in (\mathbb{R}^M)^T$. We need to show that $\{\tilde{F}[n]\}_{n=1}^{\infty}$ converges in distribution to $\tilde{F}$ continuously, i.e. that the sequence of random variables $\{\tilde{F}(X[n])[n]\}_{n=1}^{\infty}$ converges in distribution to $\tilde{F}(X)$ and thus

$$\{f^{(D+1)}(x_t[n])[n]\}_{t=1}^{T} \xrightarrow{D} \{f^{(D+1)}(x_t)\}_{t=1}^{T}. \tag{19}$$

We will do so by establishing $\{f^{(\mu)}(x_t[n])[n]\}_{t=1}^{T} \xrightarrow{D} \{f^{(\mu)}(x_t)\}_{t=1}^{T}$ inductively for every $\mu \in \{1, \ldots, D+1\}$.

For the case $\mu = 1$, let $\alpha \in \mathbb{R}^{T \times |\mathbb{N}|}$ with finite support $\{1, \ldots, T\} \times I$.

By definition (Eq. (6.1)), it is straightforward to verify that

$$f^{(1)}(X[n], \alpha)[n] \sim \mathcal{N}(0, \alpha^\top K^{(1)}(X[n], X[n])\alpha)$$
$$f^{(1)}(X, \alpha) \sim \mathcal{N}(0, \alpha^\top K^{(1)}(X, X)\alpha).$$

Let c be a continuity point of $f^{(1)}(X, \alpha)$ so that $c \neq 0$ if $\alpha^\top K^{(1)}(X, X)\alpha = 0$. Extend the CDF Φ of the standard normal distribution by setting $\Phi(-\infty) = 0$ and $\Phi(\infty) = 1$. Then the map

$$z \mapsto \Phi\left(\frac{c}{\sqrt{z}}\right) \tag{20}$$

is continuous on $(0, \infty)$ and is right-continuous at $z = 0$ if $c \neq 0$. Now since the kernel is a continuous function (Lemma .9) and since $X[n] \to X$, then $\alpha^\top K^{(1)}(X[n], X[n])\alpha \to \alpha^\top K^{(1)}(X, X)\alpha$. Moreover, since we just established that the map given by Eq. (20) is continuous, then it follows that

$$\Phi\left(\frac{c}{\sqrt{\alpha^\top K^{(1)}(X[n], X[n])\alpha}}\right) \to \Phi\left(\frac{c}{\sqrt{\alpha^\top K^{(1)}(X, X)\alpha}}\right) \text{ as } n \to \infty,$$

and hence $f^{(1)}(X[n], \alpha)[n] \xrightarrow{D} f^{(1)}(X, \alpha)$. By the Cramér-Wold Device, we deduce that $f_I^{(1)}(X[n])[n] \xrightarrow{D} f_I^{(1)}(X)$.

Now suppose that $f_I^{(\mu)}(X[n])[n] \xrightarrow{D} f_I^{(\mu)}(X)$ for every finite subset $I \subseteq \mathbb{N}$ and for some $\mu \in \{1, \ldots, D\}$. We then want to show that this same convergence holds for $\mu + 1$. Let

$\alpha \in \mathbb{R}^{T\times|\mathbb{N}|}$ with finite support $\{1,\dots,T\}\times I$. We view α as a block vector where α_{ti} is the t-th element in the i-th block. By Lemmas .14-.16, the sequence of summands $\{\gamma_j^{(\mu+1)}(X[n],\alpha)\}_{j=1}^{H_\mu[n]}$ for $n\in\mathbb{N}$ satisfies the conditions of Thm. .13; condition 1 is immediate since the weights $w_{i1}^{(\mu+1)}$ and $w_{j2}^{(\mu+1)}$ are independent and have mean 0. Theorem .13 then tells us that the projections $S^{(\mu+1)}(X[n],\alpha)[n] \xrightarrow{D} \mathcal{N}(0,\sigma^2(X,\alpha))$, where the limiting variance is given by $\sigma^2_{(\mu+1)}(X,\alpha)$ (Eq. (12)). By the Cramér-Wold Device, this implies

$$f_I^{(\mu+1)}(X[n])[n] - b_I^{(\mu+1)}\otimes \mathbf{1}_T \xrightarrow{D} \mathcal{N}(0, v_w L_{II}^{(\mu)}(X,X)),$$

which in turn implies

$$F_I^{(\mu+1)}(X[n])[n] \xrightarrow{D} F_I^{(\mu+1)}(X) \sim \mathcal{N}(0, K_{II}^{(\mu+1)}(X,X)).$$

Equation (19) then follows by induction, thus establishing continuous distributional convergence. □

Next is the proof of the bottleneck NNGP theorem, which is the main theorem of our paper.

Proof of Thm. 6.4. We proceed by induction on $d\in\{1,\dots,D\}$. The case $d=1$ is given to us by Thm. 6.3 (i.e., no hidden bottlenecks). Now suppose the claim holds for some $d\in\{1,\dots,D-1\}$. We will prove the claim for the case $d+1$.

Let $X=\{x_t\}_{t=1}^T$ be a finite subset of $\mathcal{X}$. Define the random variables

$$Z[n] = \{(F^{(d)}[n]\circ\cdots\circ F^{(1)}[n])(x_t)\}_{t=1}^T,$$
$$Z = \{(F^{(d)}\circ\cdots\circ F^{(1)})(x_t)\}_{t=1}^T.$$

Let $\tilde{F}^{(d+1)}[n]$ (resp. $\tilde{F}^{(d+1)}$) be the batch BNN (resp. batch NNGP) corresponding to $F^{(d+1)}[n]$ (resp. $F^{(d+1)}$), and observe that

$$\tilde{F}^{(d+1)}(Z[n])[n] = \{(F^{(d+1)}[n]\circ\cdots\circ F^{(1)}[n])(x_t)\}_{t=1}^T,$$
$$\tilde{F}^{(d+1)}(Z) = \{(F^{(d+1)}\circ\cdots\circ F^{(1)})(x_t)\}_{t=1}^T.$$

We proceed to establish the four conditions of Lemma .7 in order to prove

$$\tilde{F}^{(d+1)}(Z[n])[n] \xrightarrow{D} \tilde{F}^{(d+1)}(Z). \tag{21}$$

By the inductive hypothesis, $F^{(d)}[n]\circ\cdots\circ F^{(1)}[n] \xrightarrow{D} F^{(d)}\circ\cdots\circ F^{(1)}$ in $((\mathbb{R}^L)^\infty,\mathcal{A})$ and thus in particular $Z[n]\xrightarrow{D} Z$, establishing condition (a). Observe that

$$\mathrm{E}[\|Z[n]\|^2] = \sum_{t=1}^T \mathrm{E}[\|(f^{(d)}[n]\circ\cdots\circ F^{(1)}[n])(x_t)\|^2].$$

Since a composition of BNNs is still a BNN (with some hidden layers having linear activation), then we can apply Lemma .8 to each expectation in the sum to get the bound

$$\mathrm{E}[\|Z[n]\|^2] \le \sum_{t=1}^T (A_t + B_t\|x_t\|^2),$$

for some constants $A_t, B_t > 0$. In other words, the sequence of second moments of $\{Z[n]\}_{n=1}^{\infty}$ is bounded, establishing condition (b). Lemma .9 gives us condition (c). Finally, Lemma .10 tells us that $\tilde{F}^{(d+1)}[n] \xrightarrow{CD} \tilde{F}^{(d+1)}$. By Prop. .5, we immediately have $\tilde{F}^{(d+1)}[n] \xrightarrow{UD} \tilde{F}^{(d+1)}$ on every compact subset of $\mathbb{R}^{T \times B_d}$, establishing condition (d).

Having verified its four conditions, Lemma .7 implies Eq. (21) and hence

$$\{(F^{(d+1)}[n] \circ \cdots \circ F^{(1)}[n])(x_t)\}_{t=1}^{T} \xrightarrow{D} \{(F^{(d+1)} \circ \cdots \circ F^{(1)})(x_t)\}_{t=1}^{T}.$$

Since this holds for any T inputs in $\mathcal{X}$, then by Thm. .1 the desired convergence in $((\mathbb{R}^L)^{\infty}, \mathcal{A})$ follows. □

Remark .11 (Nonlinear bottleneck)**.** *Theorem 6.4 holds even if we replace $F^{(d)}[n]$ and $F^{(d)}$ with $F^{(d)}[n] \circ \left(\frac{1}{\sqrt{B_{d-1}}}\phi\right)$ and $F^{(d)} \circ \left(\frac{1}{\sqrt{B_{d-1}}}\phi\right)$ respectively for $d \in \{2, \ldots, D\}$. The proof is nearly identical, making the necessary replacements where appropriate. The only additional step needed is to verify condition (c) of Lemma .7 for $\tilde{F}^{(d+1)}[n] \circ \left(\frac{1}{\sqrt{B_d}}\phi\right)$ in the inductive step; by Lemma .7, $\tilde{F}^{(d+1)}[n] \xrightarrow{UD} \tilde{F}^{(d+1)}$ and hence $\tilde{F}^{(d+1)}[n] \xrightarrow{CD} \tilde{F}^{(d+1)}$ by Prop. .5. Now since $x \mapsto \frac{1}{\sqrt{B_d}}\phi(x)$ is (sequentially) continuous, then $\tilde{F}^{(d+1)}\left(\frac{1}{\sqrt{B_d}}\phi(x_n)\right) \to \tilde{F}^{(d+1)}\left(\frac{1}{\sqrt{B_d}}\phi(x)\right)$ whenever $x_n \to x$. By Prop. .5, $\tilde{F}^{(d+1)}[n] \circ \left(\frac{1}{\sqrt{B_d}}\phi\right) \xrightarrow{UD} \tilde{F}^{(d+1)} \circ \left(\frac{1}{\sqrt{B_d}}\phi\right)$, establishing condition (c).*

Remark .12 (Discontinuous nonlinearity)**.** *Theorem 6.4 holds even if the nonlinearity $\phi : \mathbb{R} \mapsto \mathbb{R}$ is continuous only almost everywhere (AE), as long as ϕ is continuous at 0 or $v_b > 0$. If ϕ is continuous AE, then the pointwise convergence in Eq. (18) holds AE, which is still sufficient for the Dominated Convergence Theorem. Moreover, the Continuous Mapping Theorem used in Lemmas .14-.16 is still applicable as long as the set of discontinuities of ϕ has measure 0 with respect to the distribution of the NNGP preactivation $f_i^{(\mu)}(x)$. If $v_b = 0$, then it becomes possible for the distribution of $f_i^{(\mu)}(x)$ to degenerate to a delta distribution concentrated at 0; if ϕ is also discontinuous at 0, then its set of discontinuities will have measure 1 with respect to the delta distribution, hence the requirement that $v_b > 0$ if ϕ is discontinuous at 0.*

G.3 Verifying the conditions of the CLT for exchangeable processes

The results in this section serve to support the proof of Lemma .10. Since Lemma .10 is similar to Lemma 12 in Matthews et al., 2018, then the results in this section are also similar to results in Matthews et al., 2018. The approach to proving Lemma .10 is to show that in the (no-bottleneck) NNGP limit, if the preactivations into one hidden layer converge in distribution continuously to a GP, then so do the preactivations into the next hidden layer. This is done using a special central limit theorem. The challenge is that the preactivations into any hidden layer after the first hidden layer are independent only in the wide limit. Moreover, the distribution of each preactivation changes as the preceding hidden layer grows in width. The following is a central limit theorem adapted specifically for this case; it is

a restatement of Lemma 10 in Matthews et al., 2018, which is in turn an adaptation of a central limit theorem for exchangeable processes by Blum et al. [18].

Theorem .13 (CLT for sequences of exchangeable sequences [Matthews et al., 2018])**.** *For each positive integer n, let $\{X_i[n]\}_{i=1}^{\infty}$ be an exchangeable sequence of random variables with mean 0, variance $\sigma^2[n]$, and finite absolute third moment. Suppose also that the variances converge to the limit $\lim_{n\to\infty}\sigma^2[n]=\sigma^2$. If*

(a) $E\{X_1[n]X_2[n]\}=0$,

(b) $\lim_{n\to\infty}E\{X_1[n]^2X_2[n]^2\}=\sigma^4$, *and*

(c) $E\{|X_1[n]|^3\}=o(\sqrt{n})$

then for any strictly increasing sequence H, the sequence of standardized partial sums $\{S[n]\}_{n=1}^{\infty}$ with

$$S[n]=\frac{1}{\sqrt{H[n]}}\sum_{i=1}^{H[n]}X_i[n]$$

converges in distribution to $\mathcal{N}(0,\sigma^2)$, where $\mathcal{N}(0,0)$ is interpreted as the constant 0.

We will apply Thm. .13 to the summands $\gamma_j^{(\mu)}(X[n],\alpha)[n]$ (Eq. (9)) to show that the projection $S^{(\mu)}(X[n],\alpha)[n]$ (Eq. (10)) converges to a GP. This requires us to verify the conditions of Thm. .13. We verify the existence of the limit $\lim_{n\to\infty}\sigma^2[n]=\sigma^2$ first. The following lemma is analogous to Lemma 11 in Matthews et al., 2018. The main difference is that the batch input X is replaced with a convergent sequence of input batches $\{X[n]\}_{n=1}^{\infty}$. We maintain the notation introduced in Sec. G.1.

Lemma .14. *Suppose that $f_I^{(\mu)}(X[n])[n]\xrightarrow{D}f_I^{(\mu)}(X)$ for some $\mu\in\{1,\dots,D\}$, and for every finite set $I\subset\mathbb{N}$. Then*

$$\lim_{n\to\infty}\sigma^2_{\mu+1}(X[n],\alpha)[n]=\sigma^2_{\mu+1}(X,\alpha),$$

where these variances are defined in Eqs. (11)-(12).

Proof. It is clear that $\mathrm{E}[\gamma_j^{(\mu+1)}(X[n],\alpha)[n]]=0$ since the weights $w_{ij}^{(\mu+1)}$ have 0 mean. We therefore have

$$\begin{aligned}
\sigma^2_{\mu+1}(X[n],\alpha)[n] &= \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2]\\
&= \mathrm{E}\left[\left(\sqrt{H_\mu[n]}\sum_{t=1}^{T}\sum_{i\in I}\alpha_{ti}w_{i1}^{(\mu)}g_1^{(\mu)}(x_t[n])[n]\right)^2\right]\\
&= H_\mu[n]\sum_{t,u=1}^{T}\sum_{i,j\in I}\alpha_{ti}\alpha_{uj}\,\mathrm{E}[w_{i1}^{(\mu)}w_{j1}^{(\mu)}]\,\mathrm{E}[g_1^{(\mu)}(x_t[n])[n]g_1^{(\mu)}(x_u[n])[n]]\\
&= v_w\sum_{t,u=1}^{T}\sum_{i,j\in I}\alpha_{ti}\alpha_{uj}\delta_{ij}\,\mathrm{E}[g_1^{(\mu)}(x_t[n])[n]g_1^{(\mu)}(x_u[n])[n]].
\end{aligned}$$

Theorem 3.5 in Billingsley [16] tells us that a limit can be moved inside an expectation operator if the sequence inside the expectation converges in distribution and is uniformly integrable. Since the preactivations $f_1^{(\mu)}(X[n])[n]$ converge in distribution and since the nonlinearity ϕ and multiplication mapping $\mathbb{R}^2$ to $\mathbb{R}$ are continuous functions, then the Continuous Mapping Theorem implies that the products of activations in the above expectations also converge in distribution. Uniform integrability holds by Cor. .22. We therefore have the limit

$$\begin{aligned}
\lim_{n\to\infty} \sigma_{\mu+1}^2(X[n],\alpha)[n] &= v_w \sum_{t,u=1}^{T} \sum_{i,j\in I} \alpha_{ti}\alpha_{uj}\delta_{ij}\, \mathrm{E}[g_1^{(\mu)}(x_t)g_1^{(\mu)}(x_u)] \\
&= v_w \sum_{t,u=1}^{T} \sum_{i,j\in I} \alpha_{ti}\alpha_{uj} L_{11}^{(\mu)}(x_t,x_u) \\
&= v_w \alpha^\top L^{(\mu)}(X,X)\alpha \\
&= \sigma_{(\mu+1)}^2(X,\alpha),
\end{aligned}$$

completing the proof. □

Condition (a) of Thm. .13 is easily verified directly in the proof of Lemma .10. We thus move to condition (b). The following lemma is analogous to Lemma 15 in Matthews et al., 2018.

Lemma .15. *Suppose that* $f_I^{(\mu)}(X[n])[n] \xrightarrow{D} f_I^{(\mu)}(X)$ *for some* $\mu \in \{1,\dots,D\}$. *and for every finite set* $I \subset \mathbb{N}$. *Then*

$$\lim_{n\to\infty} \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2 \gamma_2^{(\mu+1)}(X[n],\alpha)[n]^2] = \sigma_\mu^4(X,\alpha).$$

Proof. We proceed in direct analogy to the proof of Lemma .14. We have

$$\begin{aligned}
&\mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2 \gamma_2^{(\mu+1)}(X[n],\alpha)[n]^2] \\
&= \mathrm{E}\left[\left(\sqrt{H_\mu[n]}\sum_{r=1}^{T}\sum_{i\in I} w_{i1}^{(\mu)} g_1^{(\mu)}(x_r[n])[n]\right)^2 \left(\sqrt{H_\mu[n]}\sum_{t=1}^{T}\sum_{k\in I} w_{k2}^{(\mu)} g_2^{(\mu)}(x_t[n])[n]\right)^2\right] \\
&= H_\mu^2[n] \sum_{r,s,t,u=1}^{T} \sum_{i,j,k,\ell\in I} \Big(\alpha_{ri}\alpha_{sj}\alpha_{tk}\alpha_{u\ell} \cdot \mathrm{E}[w_{i1}^{(\mu)} w_{j1}^{(\mu)}] \cdot \mathrm{E}[w_{k2}^{(\mu)} w_{\ell 2}^{(\mu)}] \\
&\cdot \mathrm{E}[g_1^{(\mu)}(x_r[n])[n] g_1^{(\mu)}(x_s[n])[n] g_2^{(\mu)}(x_t[n])[n] g_2^{(\mu)}(x_u[n])[n]]\Big) \\
&= v_w^2 \sum_{r,s,t,u=1}^{T} \sum_{i,j,k,\ell\in I} \Big(\alpha_{ri}\alpha_{sj}\alpha_{tk}\alpha_{u\ell}\delta_{ij}\delta_{k\ell} \\
&\cdot \mathrm{E}[g_1^{(\mu)}(x_r[n])[n] g_1^{(\mu)}(x_s[n])[n] g_2^{(\mu)}(x_t[n])[n] g_2^{(\mu)}(x_u[n])[n]]\Big).
\end{aligned}$$

Since the preactivations $f_I^{(\mu)}(X[n])[n]$ converge in distribution for $I = \{1,2\}$, and since the nonlinearity ϕ and multiplication from $\mathbb{R}^4$ to $\mathbb{R}$ are continuous functions, then the Continuous Mapping Theorem implies the four-way products of activations in each expectation above

converge in distribution as well. Corollary .21 also tells us that the set of these four-way products of activations is uniformly integrable. By Theorem 3.5 in Billingsley [16], we have the limit

$$\begin{aligned}&\lim_{n\to\infty} \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2\gamma_2^{(\mu+1)}(X[n],\alpha)[n]^2]\\&= v_w^2 \sum_{r,s,t,u=1}^{T} \sum_{i,j,k,\ell\in I} \alpha_{ri}\alpha_{sj}\alpha_{tk}\alpha_{u\ell}\delta_{ij}\delta_{k\ell}\,\mathrm{E}[g_1^{(\mu)}(x_r)g_1^{(\mu)}(x_s)g_2^{(\mu)}(x_t,\alpha)g_2^{(\mu)}(x_u,\alpha)].\end{aligned} \tag{22}$$

Since parallel activations in a layer decorrelate in an NNGP, then we have

$$\begin{aligned}&\lim_{n\to\infty} \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2\gamma_2^{(\mu+1)}(X[n],\alpha)[n]^2]\\&= v_w^2 \sum_{r,s,t,u=1}^{T} \sum_{i,j,k,\ell\in I} \alpha_{ri}\alpha_{sj}\alpha_{tk}\alpha_{u\ell}\delta_{ij}\delta_{k\ell}\,\mathrm{E}[g_1^{(\mu)}(x_r)g_1^{(\mu)}(x_s)]\,\mathrm{E}[g_2^{(\mu)}(x_t)g_2^{(\mu)}(x_u)]\\&= \left(v_w \sum_{r,s=1}^{T} \sum_{i,j\in I} \alpha_{ri}\alpha_{sj}\delta_{ij}\,\mathrm{E}[g_1^{(\mu)}(x_r)g_1^{(\mu)}(x_s)]\right)\\&\cdot \left(v_w \sum_{t,u=1}^{T} \sum_{k,\ell\in I} \alpha_{tk}\alpha_{u\ell}\delta_{k\ell}\,\mathrm{E}[g_2^{(\mu)}(x_t)g_2^{(\mu)}(x_u)]\right)\\&= v_w\alpha^\top L^{(\mu)}(X,X)\alpha v_w\alpha^\top L^{(\mu)}(X,X)\alpha\\&= \sigma^4_{(\mu+1)}(X,\alpha),\end{aligned}$$

completing the proof. □

Finally, we verify condition (c) of Thm. .13. The following lemma is analogous to Lemma 16 in Matthews et al., 2018.

Lemma .16. *Suppose that* $f_I^{(\mu)}(X[n])[n] \xrightarrow{D} f_I^{(\mu)}(X)$ *for some* $\mu \in \{1,\dots,D\}$*, and for every finite set* $I \subset \mathbb{N}$*. Then*

$$\mathrm{E}[|\gamma_1^{(\mu+1)}(X[n],\alpha)[n]|^3] = o(\sqrt{n}).$$

Proof. We will prove the stronger result that the third absolute moment of $\gamma_1^{(\mu+1)}(X[n],\alpha)[n]$ is bounded over n. By Hölder's inequality,

$$\mathrm{E}[|\gamma_1^{(\mu+1)}(X[n],\alpha)[n]|^3] \le \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^4]^{\frac{3}{4}}.$$

Thus, to bound the left side independently of n, it is sufficient to do the same for the fourth moment of $\gamma_1^{(\mu+1)}(X[n],\alpha)[n]$. Observe that

$$\mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^4]^{\frac{3}{4}} = \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^2],$$

where the right-hand side is similar to the quantity discussed in Lemma .15. Therefore, calculations proceed in direct analogy to the proof of Lemma .15 up to and including Eq. (22).

Thus, we have

$$\lim_{n\to\infty} \mathrm{E}[\gamma_1^{(\mu+1)}(X[n],\alpha)[n]^4]^{\frac{3}{4}}$$
$$= v_w^2 \sum_{r,s,t,u=1}^{T} \sum_{i,j,k,\ell\in I} \alpha_{ri}\alpha_{sj}\alpha_{tk}\alpha_{u\ell}\delta_{ij}\delta_{k\ell}\,\mathrm{E}[g_1^{(\mu)}(x_r,\alpha)g_1^{(\mu)}(x_s,\alpha)g_1^{(\mu)}(x_t,\alpha)g_1^{(\mu)}(x_u,\alpha)].$$

The right-hand side can be shown to be finite by applying Lemma .18 to bound the expectation of the four-way product by a product of eighth moments, applying the linear envelope property to obtain bounds in terms of preactivations, and finally noting that the eighth moment of a normal distribution is finite; this gives us the desired bound on the fourth and hence third absolute moment. □

G.4 Establishing uniform integrability

The results in this section serve to support the proofs in Appendix G.3. As in Appendix G.3, the results in this appendix are stronger versions of results appearing in Matthews et al., 2018. The key results in this section are Lemma .18 and Cors. .21 and .22 and are the only ones referenced outside of this section.

Lemma .17. *Let X be a random variable. Then $E[X^4] \leq E[X^8]^{\frac{1}{2}}$.*

Proof. By Hölder's Inequality, we have

$$E[X^4] = E[X^4 \cdot 1] \leq E[(X^4)^2]^{\frac{1}{2}} E[1^2]^{\frac{1}{2}} = E[X^8]^{\frac{1}{2}}.$$

□

The following lemma is a stronger version of Lemma 18 in Matthews et al., 2018. Matthews et al., 2018 proves that the expectation $E\left[\prod_{i=1}^4 |X_i|^{p_i}\right]$ is uniformly bounded by a polynomial in the eighth moments $E[X_i^8] < \infty$ for $i \in \{1,2,3,4\}$ without specifying the polynomial. Lemma .18 below provides the explicit bound $\prod_{i=1}^4 E[X_i^8]^{\frac{p_i}{8}}$, which is a polynomial in the eighth moments. This bound is important when proving uniform convergence with respect to the inputs of a random neural network, since the coefficients and exponents in the bound are independent of the network's input.

Lemma .18. *Let X_i be random variables with $E[X_i^8] < \infty$ for $i \in \{1,2,3,4\}$. Then for any choice of $p_i \in \{0,1,2\}$ it holds that*

$$E\left[\prod_{i=1}^4 |X_i|^{p_i}\right] \leq \prod_{i=1}^4 E[X_i^8]^{\frac{p_i}{8}}.$$

Proof. Using Hölder's inequality twice, we have

$$\begin{aligned}
E[|X_1|^{p_1}|X_2|^{p_2}|X_3|^{p_3}|X_4|^{p_4}] &\leq E[(|X_1|^{p_1}|X_2|^{p_2})^2]^{\frac{1}{2}} E[(|X_3|^{p_3}|X_4|^{p_4})^2]^{\frac{1}{2}} \\
&= E[X_1^{2p_1} X_2^{2p_2}]^{\frac{1}{2}} E[X_3^{2p_3} X_4^{2p_4}]^{\frac{1}{2}} \\
&\leq \left(E[(X_1^{2p_1})^2]^{\frac{1}{2}} E[(X_2^{2p_2})^2]^{\frac{1}{2}}\right)^{\frac{1}{2}} \left(E[(X_3^{2p_3})^2]^{\frac{1}{2}} E[(X_4^{2p_4})^2]^{\frac{1}{2}}\right)^{\frac{1}{2}} \\
&= E[X_1^{4p_1}]^{\frac{1}{4}} E[X_2^{4p_2}]^{\frac{1}{4}} E[X_3^{4p_3}]^{\frac{1}{4}} E[X_4^{4p_4}]^{\frac{1}{4}} \\
&= \prod_{i=1}^{4} E[X_i^{4p_i}]^{\frac{1}{4}}.
\end{aligned}$$

If $p_i = 0$, then $E[X_i^{4p_i}]^{\frac{1}{4}} = E[1]^{\frac{1}{4}} = 1$, which can be written as $E[X_i^8]^0$. If $p_i = 1$, then by Lemma .17, $E[X_i^{4p_i}]^{\frac{1}{4}} \leq E[X_i^8]^{\frac{1}{8}}$. If $p_i = 2$, then we simply have $E[X_i^{4p_i}]^{\frac{1}{4}} = E[X_i^8]^{\frac{2}{8}}$. We therefore see that for any $p_i \in \{0, 1, 2\}$, $E[X_i^{4p_i}]^{\frac{1}{4}} = E[X_i^8]^{\frac{p_i}{8}}$. Substituting this into the above product yields the desired bound. □

The following lemma extends Lemma 20 in Matthews et al., 2018 to stochastic processes in the sense that the input into the BNN is now a variable. We can achieve a uniform bound if we assume that the input space is compact.

Lemma .19. *Let $\mathcal{X} \subset \mathbb{R}^M$ be a compact input space. Then for each $\mu \in \{1, \ldots, D+1\}$, the eighth moments of the normally distributed random variables $f_i^{(\mu)}(x)[n]$ defined by equation (6.1) are uniformly bounded over all $i \in \{1, \ldots, h_\mu(n)\}$, $n \in \mathbb{N}$ and $x \in \mathcal{X}$.*

Proof. We proceed by induction on μ. The case $\mu = 1$ is trivial; the random variables $f_i^{(1)}(x)[n]$ are IID over i and follow the normal distribution $\mathcal{N}(0, v_b^{(1)} + v_w^{(1)}\|x\|^2)$. The eighth moments are therefore

$$E[f_i^{(1)}(x)[n]^8] = 105(v_b^{(1)} + v_w^{(1)}\|x\|^2)^4.$$

Clearly the eighth moment is independent of i and n. Moreover, since $\mathcal{X}$ is compact, then $\sup_{x\in\mathcal{X}} E[f_i^{(1)}(x)[n]^8] < \infty$. The eighth moments are therefore uniformly bounded over i, n and x.

Now assume that the eighth moments of $f_i^{(\mu)}(x)[n]$ are uniformly bounded over i, n and x for all $\mu \in \{1, \ldots, t-1\}$ and for some $t \in \{2, \ldots, D+1\}$. We wish to prove that the eighth moments of $f_i^{(t)}(x)[n]$ are uniformly bounded over i, n and x. Using the inequality $|u(x) + v(x)|^p \leq 2^{p-1}(|u(x)|^p + |v(x)|^p)$ for elements u and v of the L^p space for $p \geq 1$, which follows from the convexity of $h(x) := x^p$ for $p > 1$, the bound

$$E[f_i^{(t)}(x)[n]^8] \leq 2^7 E\left[(b_i^{(t)})^8 + \left(\sum_{j=1}^{h_{t-1}(n)} w_{ij}^{(t)} g_j^{(t)}(x)[n]\right)^8\right]$$

is first established. The term $E[(b_i^{(t)})^8]$ is bounded since the biases are normally distributed. Moreover, the biases are IID over i and are independent of n. Therefore, to achieve the desired uniform bound, we only need to show that the term

$$S_i(x)[n] := E\left[\left(\sum_{j=1}^{h_{t-1}(n)} w_{ij}^{(t)} g_j^{(t-1)}(x)[n]\right)^8\right]$$

is uniformly bounded over i, n and x. By Lemma 20 in Matthews et al., 2018,

$$S_i(x)[n] \leq \frac{1}{h_{t-1}(n)^4} E\left[\left(\sum_{i=1}^{h_{t-1}(n)} (c^2 + 2cm|f_i^{(t-1)}(x)[n]| + m^2|f_i^{(t-1)}(x)[n]|^2)\right)^4\right],$$

where $c, m > 0$ are constants from the linear envelope property of the activation function. Letting $a = \max\{c^2, 2cm, m^2\}$ and multiplying out the quantity in the above expectation, we have

$$\begin{aligned} S_i(x)[n] &\leq \frac{a^4}{h_{t-1}(n)^4} E\left[\sum_{i,j,k,\ell=1}^{h_{t-1}(n)} \sum_{p,q,r,s=0}^{2} |f_i^{(t-1)}(x)[n]|^p \cdot |f_j^{(t-1)}(x)[n]|^q \right. \\ &\quad \left. \cdot |f_k^{(t-1)}(x)[n]|^r \cdot |f_\ell^{(t-1)}(x)[n]|^s\right] \\ &= \frac{a^4}{h_{t-1}(n)^4} \sum_{i,j,k,\ell=1}^{h_{t-1}(n)} \sum_{p,q,r,s=0}^{2} E\left[|f_i^{(t-1)}(x)[n]|^p \cdot |f_j^{(t-1)}(x)[n]|^q \right. \\ &\quad \left. \cdot |f_k^{(t-1)}(x)[n]|^r \cdot |f_\ell^{(t-1)}(x)[n]|^s\right]. \end{aligned}$$

Using Lemma .18 and the fact that the moments of $f_i^{(t-1)}(x)[n]$ are independent of i by exchangeability, we have

$$\begin{aligned} S_i(x)[n] &\leq \frac{a^4}{h_{t-1}(n)^4} \sum_{i,j,k,\ell=1}^{h_{t-1}(n)} \sum_{p,q,r,s=0}^{2} \left(E[f_i^{(t-1)}(x)[n]^8]^{\frac{p}{8}} \cdot E[f_j^{(t-1)}(x)[n]^8]^{\frac{q}{8}} \right. \\ &\quad \left. \cdot E[f_k^{(t-1)}(x)[n]^8]^{\frac{r}{8}} \cdot E[f_\ell^{(t-1)}(x)[n]^8]^{\frac{s}{8}}\right) \\ &= \frac{a^4}{h_{t-1}(n)^4} \sum_{i,j,k,\ell=1}^{h_{t-1}(n)} \sum_{p,q,r,s=0}^{2} E[f_1^{(t-1)}(x)[n]^8]^{\frac{p+q+r+s}{8}} \\ &= \frac{a^4}{h_{t-1}(n)^4} \cdot h_{t-1}(n)^4 \sum_{p,q,r,s=0}^{2} E[f_1^{(t-1)}(x)[n]^8]^{\frac{p+q+r+s}{8}} \\ &= a^4 \sum_{p,q,r,s=0}^{2} E[f_1^{(t-1)}(x)[n]^8]^{\frac{p+q+r+s}{8}} \\ &= a^4 \sum_{j=1}^{81} E[f_1^{(t-1)}(x)[n]^8]^{m_j}, \end{aligned}$$

where each m_j is a rational number between 0 and 1. Define the function

$$\psi(z) = a^4 \sum_{j=1}^{81} z^{m_j},$$

and note that a and the m_j are independent of the hidden width index n, the hidden neuron index i, and the input x. Moreover, ψ is increasing on the interval $(0, \infty)$. Since we assumed as our inductive hypothesis that $E[f_i^{(t-1)}(x)[n]^8] < \infty$, then it follows that

$$\sup_{i,n,x} S_i(x)[n] \leq \psi(\sup_{i,n,x} E[f_1^{(t-1)}(x)[n]^8]) < \infty,$$

implying that $E[f_i^{(t)}(x)[n]^8] < \infty$ uniformly over i n, and x, thereby completing the proof. □

The following lemma extends Lemma 21 in Matthews et al., 2018 to stochastic processes in the same sense as Lemma .19 above.

Lemma .20. *Let $\mathcal{X} \subset \mathbb{R}^M$ be a compact input space. Then for any $\mu \in \{1, \ldots, D+1\}$ and indices $i, j, k, \ell \in \mathbb{N}$, the set of random variables*

$$S := \{g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n] g_k^{(\mu)}(x_3)[n] g_\ell^{(\mu)}(x_4)[n] : n \in \mathbb{N} \text{ and } x_1, x_2, x_3, x_4 \in \mathcal{X}\}$$

is uniformly integrable.

Proof. By the de la Vallée-Poussin Theorem [109, p.19, Theorem T22], S is uniformly integrable if

$$\sup_{n,x_1,x_2,x_3,x_4} E[|g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n] g_k^{(\mu)}(x_3)[n] g_\ell^{(\mu)}(x_4)[n]|^{1+\varepsilon}] < \infty \text{ for some } \varepsilon > 0.$$

We consider $\varepsilon = 1$. By Lemma .18,

$$\begin{aligned}
&E[|g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n] g_k^{(\mu)}(x_3)[n] g_\ell^{(\mu)}(x_4)[n]|^2] \\
&\leq E[g_i^{(\mu)}(x_1)[n]^8]^{\frac{1}{4}} E[g_j^{(\mu)}(x_2)[n]^8]^{\frac{1}{4}} E[g_k^{(\mu)}(x_3)[n]^8]^{\frac{1}{4}} E[g_\ell^{(\mu)}(x_4)[n]^8]^{\frac{1}{4}} \\
&= \prod_{q=1}^{4} E[g_1^{(\mu)}(x_q)[n]^8]^{\frac{1}{4}},
\end{aligned}$$

where we obtained the last line by exchangeability over the indices i, j, k, ℓ. We therefore have

$$\begin{aligned}
\sup_{n,x_1,x_2,x_3,x_4} E[|g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n] g_k^{(\mu)}(x_3)[n] g_\ell^{(\mu)}(x_4)[n]|^2] &\leq \prod_{q=1}^{4} \sup_{n,x_q} E[g_1^{(\mu)}(x_q)[n]^8]^{\frac{1}{4}} \\
&= \prod_{q=1}^{4} \sup_{n,x} E[g_1^{(\mu)}(x)[n]^8]^{\frac{1}{4}} \\
&= \sup_{n,x} E[g_1^{(\mu)}(x)[n]^8].
\end{aligned}$$

It thus suffices to show that the supremum in the last line is finite. By the linear envelope property of the activation function,

$$E[g_1^{(\mu)}(x)[n]^8] \leq 2^7 \left(c^8 + m^8 E[f_1^{(\mu)}(x)[n]^8]\right).$$

By Lemma .19, the right-hand side is uniformly bounded over all $n \in \mathbb{N}$ and $x \in \mathcal{X}$, completing the proof. □

Cors. .21 and .22 of Lemma .20, below, are used in Lemmas .15 and .14, respectively.

Corollary .21. *Let $\{x_q[n] \in \mathbb{R}^M\}_{n=1}^{\infty}$ for $q \in \{1, \ldots, 4\}$ be four convergent sequences with finite limits. Then for any $\mu \in \{1, \ldots, D+1\}$ and indices $i, j, k, \ell \in \mathbb{N}$, the set of random variables*

$$S = \{g_i^{(\mu)}(x_1[n])[n] g_j^{(\mu)}(x_2[n])[n] g_k^{(\mu)}(x_3[n])[n] g_\ell^{(\mu)}(x_4[n])[n] : n \in \mathbb{N}\}$$

is uniformly integrable.

Proof. Since the sequences $\{x_q[n]\}_{n=1}^{\infty}$ for $q \in \{1, \ldots, 4\}$, converge to finite limits, then there exists a compact set $\mathcal{X} \subset \mathbb{R}^M$ that contains $x_q[n]$ for all n and q. By (the proof of) Lemma .20, we have

$$\sup_{n, x_1, x_2, x_3, x_4} E[|g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n] g_k^{(\mu)}(x_3)[n] g_\ell^{(\mu)}(x_4)[n]|^2] < \infty,$$

where the x_q are elements of the compact set $\mathcal{X}$. It then holds in particular that

$$\sup_n E[|g_i^{(\mu)}(x_1[n])[n] g_j^{(\mu)}(x_2[n])[n] g_k^{(\mu)}(x_3[n])[n] g_\ell^{(\mu)}(x_4[n])[n]|^2] < \infty.$$

Uniform integrability then follows by the de la Vallee-Poussin Theorem. □

Corollary .22. *Let $\{x_q[n] \in \mathbb{R}^M\}_{n=1}^{\infty}$ for $q \in \{1, 2\}$, be two convergent sequences with finite limits. Then for any $\mu \in \{1, \ldots, D+1\}$ and indices $i, j \in \mathbb{N}$, the set of random variables*

$$S = \{g_i^{(\mu)}(x_1[n])[n] g_j^{(\mu)}(x_2[n])[n] : n \in \mathbb{N}\}$$

is uniformly integrable.

Proof. By the de la Vallée-Poussin Theorem, S is uniformly integrable if

$$\sup_{n, x_1, x_2, x_3, x_4} E[|g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n]|^{1+\varepsilon}] < \infty \text{ for some } \varepsilon > 0.$$

We consider $\varepsilon = 3$. We have

$$\begin{aligned} &\sup_{n, x_1, x_2, x_3, x_4} E[|g_i^{(\mu)}(x_1)[n] g_j^{(\mu)}(x_2)[n]|^4] \\ &= \sup_{n, x_1, x_2, x_3, x_4} E[g_i^{(\mu)}(x_1)[n]^2 g_i^{(\mu)}(x_1)[n]^2 g_j^{(\mu)}(x_2)[n]^2 g_j^{(\mu)}(x_2)[n]^2], \end{aligned}$$

which is finite by (the proof of) Cor. .21. The claim then follows. □

H Correspondence to the no-bottleneck NNGP

The following is our proof of the Wide Bottleneck Correspondence Theorem for the case of a single-bottleneck NNGP.

Proof of Thm. 6.8. First we prove statement (b). We will do so for $L = 1$; the case $L > 1$ proceeds similarly. Let $X = \{x_t\}_{t=1}^T$ be a finite set of inputs. Let $p_H : \mathbb{R}^T \mapsto \mathbb{R}$ and $p : \mathbb{R}^T \mapsto \mathbb{R}$ be the PDFs of $F^{(H)}$ and F, respectively. Let $k^{(D_1)} : \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}$ and $k^{(D_2)} : \mathbb{R}^H \times \mathbb{R}^H \mapsto \mathbb{R}$ be the NNGP kernels of the pre-bottleneck and post-bottleneck components (with respective depths D_1 and D_2) of $F^{(H)}$; note that the kernels are independent of H. Then the PDF p_H is given by

$$p_H(y) = \int_{(\mathbb{R}^T)^H} \left(\mathcal{N}\left(y; 0, k^{(D_2)}\left(\frac{1}{\sqrt{H}}\phi(\{h_i\}_{i=1}^H), \frac{1}{\sqrt{H}}\phi(\{h_j\}_{j=1}^H)\right) + v_n \mathbf{I}_T\right) \cdot \prod_{k=1}^H \mathcal{N}(h_k; 0, k^{(D_1)}(X, X)) \right) \mathrm{d}h_k, \tag{23}$$

where $h_i \in \mathbb{R}^T$ is the vector of preactivations into the i-th hidden neuron in the bottleneck, and where we use the notation $\mathcal{N}(z; \mu, \Sigma)$ to mean the normal PDF in the variable z with mean μ and covariance Σ. Observing that the NNGP kernel in Eqs. (6.6)-(6.7) depends on its inputs only through their Gram matrix and writing the kernel of the first layer explicitly, we can define a function $\tilde{k}^{(D_2)}$ on the space of symmetric positive semidefinite matrices such that

$$\tilde{k}^{(D_2)}(v_b + v_w A) = k^{(D_2)}(B, B), \quad A = BB^\top.$$

Defining the random $T \times T$ matrix

$$Z_H = \frac{1}{H}\sum_{i=1}^H \phi(h_i)\phi(h_i)^\top, \quad h_i \sim \mathcal{N}(0, k^{(D_1)}(X, X)) \text{ IID}, \tag{24}$$

and letting μ_H denote the probability measure associated with Z_H, the PDF in Eq. (23) can be written as

$$p_H(y) = \int_{\mathbb{R}^{T\times T}} \mathcal{N}(y; 0, \tilde{k}^{(D_2)}(v_b + v_w z) + v_n \mathbf{I}_T)\, \mathrm{d}\mu_H(z).$$

Here z is a dummy variable. Now since the h_i in Eq. (24) are IID, then so are the matrices $\phi(h_i)\phi(h_i)^\top$. Therefore, Z_H is an empirical average of H IID random matrices. By the Law of Large Numbers, we have

$$\{Z_H\}_{H=1}^\infty \xrightarrow{P} Z = \mathrm{E}_{h\sim\mathcal{N}(0,k^{(D_1)}(X,X))}[\phi(h)\phi(h)^\top],$$

where the convergence is in probability. In particular, $\{Z_H\}_{H=1}^\infty \xrightarrow{D} Z$ so that the sequence of measures $\{\mu_H\}_{H=1}^\infty$ weakly converges to the probability measure μ associated with Z. Note that μ is a delta distribution concentrated at Z. Furthermore, thanks to the Gaussian noise, the function $z \to \mathcal{N}(y; 0, \tilde{k}^{(D_2)}(z) + v_n \mathbf{I}_T)$ is bounded over $\mathbb{R}^{T\times T}$; it is continuous as well, as the matrix inversion and determination operations and the NNGP kernel (Lemma .9) are all

continuous. By the weak convergence of measures and the delta distribution μ, we have

$$\begin{aligned}
\lim_{H\to\infty} p_H(y) &= \lim_{H\to\infty} \int_{\mathbb{R}^{T\times T}} \mathcal{N}(y; 0, \tilde{k}^{(D_2)}(v_b + v_w z) + v_n \mathbf{I}_T)\, \mathrm{d}\mu_H(z) \\
&= \int_{\mathbb{R}^{T\times T}} \mathcal{N}(y; 0, \tilde{k}^{(D_2)}(v_b + v_w z) + v_n \mathbf{I}_T)\, \mathrm{d}\mu(z) \\
&= \mathcal{N}\left(y; 0, \tilde{k}^{(D_2)}\left(v_b + v_w \,\mathrm{E}_{h\sim\mathcal{N}(0,k^{(D_1)}(X,X)}[\phi(h)\phi(h)^\top]\right) + v_n \mathbf{I}_T\right) \\
&= \mathcal{N}\left(y; 0, \tilde{k}^{(D_2)}\left(k^{(D_1+1)}(X,X)\right) + v_n \mathbf{I}_T\right) \\
&= \mathcal{N}(y; 0, k^{(D_1+D_2+1)}(X,X) + v_n \mathbf{I}_T) \\
&= p(y),
\end{aligned}$$

which is the PDF of an NNGP with $D_1 + D_2 + 1$ hidden layers.

To prove statement (a) of the theorem, we first note that the pointwise convergence $p_H \to p$ ensures the convegence in distribution $F^{(H)}(X) \xrightarrow{D} F(X)$ according to Scheffé's Lemma. Since this holds for any finite set of inputs X and in particular any finite subset of a countable set $\mathcal{X} \subset \mathbb{R}^M$, then by Thm. .1, we have that $\{F^{(H)}\}_{H=1}^{\infty} \xrightarrow{D} F$ in $((\mathbb{R}^L)^\infty, \mathcal{A})$ for inputs restricted to $\mathcal{X}$ as claimed. □

I Bottleneck layers induce correlation

Recall the single-bottleneck NNGP F defined in Sec. 6.4.1. Each output $(F_i(x_1), F_i(x_2))$ conditional on the activations of the bottleneck layer follow the two-dimensional normal distribution $\mathcal{N}(0, K)$ where

$$K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}.$$

It can be shown that the diagonal entries of K are given by

$$\begin{aligned}
k_{aa} &= b_D + \frac{w_D}{H} \sum_{i=1}^{H} \phi(h_i^a)^2, \\
b_D &= v_n + v_b \sum_{d=0}^{D-1} v_w^d, \\
w_D &= v_w^D.
\end{aligned}$$

The expression for the off-diagonals k_{12}, k_{21} will not be important.

The proof of Prop. 6.10 regarding the quadratic correlation between bottleneck NNGP outputs follows.

Proof of Prop. 6.10. We have

$$\begin{aligned}
\mathrm{E}[F_1(x_a)^2] &= \int_{(\mathbb{R}^2)^H} \int_{(\mathbb{R}^2)^2} (y_1^a)^2 \mathcal{N}(y_1; 0, K) \mathcal{N}(y_2; 0, K) \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}y \,\mathrm{d}h \\
&= \int_{(\mathbb{R}^2)^H} k_{aa} \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}h \\
&= \int_{(\mathbb{R}^2)^H} \left(b_D + \frac{w_D}{H} \sum_{i=1}^{H} \phi(h_i^a)^2 \right) \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}h \\
&= b_D + \frac{w_D}{H} \sum_{i=1}^{H} \int_{(\mathbb{R}^2)^H} \phi(h_i^a)^2 \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}h \\
&= b_D + \frac{w_D}{H} \sum_{i=1}^{H} \int_{(\mathbb{R}^2)^H} \phi(h_i^a)^2 \mathcal{N}(h_i; 0, C) \,\mathrm{d}h_i \\
&= b_D + \frac{w_D}{H} \sum_{i=1}^{H} E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] \\
&= b_D + w_D E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2].
\end{aligned} \tag{25}$$

We similarly have

$$\mathrm{E}[F_2(x_b)^2] = b_D + w_D E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2]. \tag{26}$$

Combining Eqs. (25) and (26) yields

$$\begin{aligned}
&\mathrm{E}[F_1(x_a)^2] \,\mathrm{E}[F_2(x_b)^2] \\
&= b_D^2 + b_D w_D E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] + b_D w_D E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2] \\
&+ E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2].
\end{aligned}$$

We also have

$$
\begin{aligned}
&\mathrm{E}[F_1(x_a)^2 F_2(x_b)^2] \\
&= \int_{(\mathbb{R}^2)^H} \int_{(\mathbb{R}^2)^2} \prod_{y=y_1^a, y_2^b} y^2 \mathcal{N}(y; 0, K) \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}y \,\mathrm{d}h \\
&= \int_{(\mathbb{R}^2)^H} k_{aa} k_{bb} \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}h \\
&= \int_{(\mathbb{R}^2)^H} \prod_{h=h_i^a, h_j^b} \left(b_D + \frac{w_D}{H} \sum_{j=1}^{H} \phi(h)^2 \right) \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C) \,\mathrm{d}h \\
&= b_D^2 + b_D w_D E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] + b_D w_D E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2] \\
&+ \frac{w_D^2}{H^2} \sum_{i,j=1}^{H} \int_{(\mathbb{R}^2)^2} \phi(h_i^a)^2 \phi(h_j^b)^2 \mathcal{N}(h_i; 0, C) \mathcal{N}(h_j; 0, C) \,\mathrm{d}h_i \,\mathrm{d}h_j \\
&= \mathrm{E}[F_1(x_a)^2] \,\mathrm{E}[F_2(x_b)^2] - E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2] \\
&+ \frac{w_D^2}{H^2} \sum_{i=1}^{H} \int_{\mathbb{R}^2} \phi(h_i^a)^2 \phi(h_i^b)^2 \mathcal{N}(h_i; 0, C) \,\mathrm{d}h_i \\
&+ \frac{w_D^2}{H^2} \sum_{i \neq j=1}^{H} \int_{(\mathbb{R}^2)^2} \phi(h_i^a)^2 \phi(h_j^b)^2 \mathcal{N}(h_i; 0, C) \mathcal{N}(h_j; 0, C) \,\mathrm{d}h_i \,\mathrm{d}h_j \\
&= \mathrm{E}[F_1(x_a)^2] \,\mathrm{E}[F_2(x_b)^2] - E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2] \\
&+ \frac{w_D^2}{H} E_{(z,z') \sim \mathcal{N}(0, C)}[\phi(z)^2 \phi(z')^2] \\
&+ w_D^2 \left(1 - \frac{1}{H}\right) E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2] \\
&= \mathrm{E}[F_1(x_a)^2] \,\mathrm{E}[F_2(x_b)^2] + \frac{w_D^2}{H} E_{(z,z') \sim \mathcal{N}(0, C)}[\phi(z)^2 \phi(z')^2] \\
&- \frac{w_D^2}{H} E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] E_{z \sim \mathcal{N}(0, c_{bb})}[\phi(z)^2] \\
&= \mathrm{E}[F_1(x_a)^2] \,\mathrm{E}[F_2(x_b)^2] + \frac{w_D^2}{H} \operatorname{Cov}_{(z,z') \sim \mathcal{N}(0, C)}[\phi(z)^2, \phi(z')^2].
\end{aligned}
$$

We therefore have

$$
\operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2] = \frac{w_D^2}{H} \operatorname{Cov}_{(z,z') \sim \mathcal{N}(0, C)}[\phi(z)^2, \phi(z')^2].
$$

On the right-hand side, we have the covariance of two rectified quadratic units with respect to the Gaussian measure $\mathcal{N}(0, C)$. By the work of Cho and Saul [28] and by adjusting for differences in normalization, we have the expectations

$$
E_{z \sim \mathcal{N}(0, c_{aa})}[\phi(z)^2] = c_{aa}, \tag{27}
$$

$$
E_{(z,z') \sim \mathcal{N}(0, C)}[\phi(z)^2 \phi(z')^2] = \frac{2}{\pi} c_{aa} c_{bb} J_2(\beta), \tag{28}
$$

where $\beta = \cos^{-1}\left(\frac{c_{ab}}{\sqrt{c_{aa}c_{bb}}}\right)$ and $J_2(\beta) = 3\sin\beta\cos\beta + (\pi - \beta)(1 + 2\cos^2\beta)$. Using also the fact that $w_D = v_w^D$, we obtain

$$\begin{aligned}\operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2] &= \frac{(v_w^D)^2}{H}\left(\frac{2}{\pi}c_{aa}c_{bb}J_2(\beta) - c_{aa}c_{bb}\right) \\ &= \frac{v_w^{2D}c_{aa}c_{bb}}{H}\left(\frac{2}{\pi}J_2(\beta) - 1\right),\end{aligned}$$

establishing Eq. (6.11).

The corresponding correlation is defined as

$$q_{ab}^{\times} = \frac{\operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2]}{\sqrt{\operatorname{V}[F_1(x_a)^2]\operatorname{V}[F_2(x_b)^2]}}. \tag{29}$$

We already know the numerator on the right-hand side, but we need to calculate the variances in the denominator. Using the fact that $F_1(x_a)$ and $F_2(x_b)$ are identically (but not independently) distributed, we have

$$\begin{aligned}&\operatorname{V}[F_1(x_a)^2] \\ &= \operatorname{E}[F_1(x_a)^4] - \operatorname{E}[F_1(x_a)^2]^2 \\ &= \int_{(\mathbb{R}^2)^H}\int_{(\mathbb{R}^2)^2}(y_1^a)^4\prod_{i=1}^{2}\mathcal{N}(y_i;0,K)\prod_{m=1}^{H}\mathcal{N}(h_m;0,C)\,\mathrm{d}y\,\mathrm{d}h - \operatorname{E}[F_1(x_a)^2]^2 \\ &= 3\int_{(\mathbb{R}^2)^H}k_{aa}k_{aa}\prod_{m=1}^{H}\mathcal{N}(h_m;0,C)\,\mathrm{d}h - \operatorname{E}[F_1(x_a)^2]^2 \\ &= 3\operatorname{E}[F_1(x_a)^2, F_2(x_a)^2] - \operatorname{E}[F_1(x_a)^2]\operatorname{E}[F_2(x_a)^2] \\ &= 3\operatorname{Cov}[F_1(x_a)^2, F_2(x_a)^2] + 3\operatorname{E}[F_1(x_a)^2]\operatorname{E}[F_2(x_a)^2] - \operatorname{E}[F_1(x_a)^2]\operatorname{E}[F_2(x_a)^2] \\ &= 3\operatorname{Cov}[F_1(x_a)^2, F_2(x_a)^2] + 2\operatorname{E}[F_1(x_a)^2]\operatorname{E}[F_2(x_a)^2] \\ &= 3\operatorname{Cov}[F_1(x_a)^2, F_2(x_a)^2] + 2\operatorname{E}[F_1(x_a)^2]^2.\end{aligned}$$

By Eqs. (25), (27), and (6.11), we have

$$\begin{aligned}\operatorname{V}[F_1(x_a)^2] &= \frac{3w_D^2c_{aa}c_{aa}}{H}\left(\frac{2}{\pi}J_2(0) - 1\right) + 2\left(b_D + w_Dc_{aa}\right)^2 \\ &= \frac{3w_D^2(c_{aa})^2}{H}\left(\frac{2\cdot 3\pi}{\pi} - 1\right) + 2\left(b_D + w_Dc_{aa}\right)^2 \\ &= \frac{15w_D^2(c_{aa})^2}{H} + 2\left(b_D + w_Dc_{aa}\right)^2.\end{aligned} \tag{30}$$

We similarly have

$$\operatorname{V}[F_2(x_b)^2] = \frac{15w_D^2(c_{bb})^2}{H} + 2\left(b_D + w_Dc_{bb}\right)^2.$$

Substituting these variances into Eq. (29), we obtain

$$\begin{aligned} q_{ab}^{\times} &= \frac{\frac{w_D^2 c_{aa} c_{bb}}{H}\left(\frac{2}{\pi}J_2(\beta)-1\right)}{\prod\limits_{c=c_{aa},c_{bb}}\sqrt{\frac{15w_D^2c^2}{H}+2\left(b_D+w_Dc\right)^2}} \\ &= \frac{\left(\frac{2}{\pi}J_2(\beta)-1\right)}{\prod\limits_{c=c_{aa},c_{bb}}\sqrt{\frac{H}{w_Dc^2}\left(\frac{15w_D^2c^2}{H}+2\left(b_D+w_Dc\right)^2\right)}} \\ &= \frac{\left(\frac{2}{\pi}J_2(\beta)-1\right)}{\prod\limits_{c=c_{aa},c_{bb}}\sqrt{15+2H\left(\frac{r_D}{c}+1\right)^2}}, \end{aligned}$$

where

$$\begin{aligned} r_D &= \frac{b_D}{w_D} \\ &= \frac{v_n}{v_w^D}+\frac{v_b}{v_w^D}\sum_{d=0}^{D-1}v_w^d \\ &= \frac{v_n}{v_w^D}+v_b\sum_{d=1}^{D}\frac{1}{v_w^d} \\ &= \begin{cases} v_n+Dv_b & \text{if } v_w=1 \\ \frac{v_n}{v_w^D}+\frac{v_b}{1-v_w}\left(\frac{1}{v_w^D}-1\right) & \text{otherwise,} \end{cases} \end{aligned}$$

establishing Eqs. (6.12) and (6.13). □

Proof of Prop. 6.12. For part (a), Eq. (6.16) follows by substitution of Eq. (6.15) into Eq. (6.12).

For part (b), the map $G \mapsto Q^{\times(\infty)}$ is a composition of the maps $G \mapsto C$ and $C \mapsto Q^{\times(\infty)}$, and thus it suffices to show that these two maps are invertible. The map $G \mapsto C$ sends the input Gram matrix to the NNGP kernel at the bottleneck layer. Inverting Eq. (6.9) for the case $i = j$, we obtain the recursion for the backward propagation of the NNGP kernel:

$$K_{ii}^{(\mu-1)}(x_1,x_2) = \frac{1}{v_w}\left(\prod_{a=1,2}\sqrt{K_{ii}^{(\mu)}(x_a,x_a)-v_b}\right)\cos J_1^{-1}\left(\pi\frac{K_{ii}^{(\mu)}(x_1,x_2)-v_b}{\prod_{a=1,2}\sqrt{K_{ii}^{(\mu)}(x_a,x_a)-v_b}}\right),$$

where we note J_1 is strictly decreasing on $[0, \pi]$. Applying this recursion to C d times (where d is the depth of the pre-bottleneck NNGP) gives $K_{ii}^{(1)}(x_1, x_2)$, and by solving Eq. (6.6) we obtain G. Thus, $G \mapsto C$ is invertible.

To show $C \mapsto Q^{\times(\infty)}$ is invertible, we inspect Eq. (6.16) for the case $v_w > 1$ in Prop. 6.12 and observe that $q_{aa}^{\times(\infty)}$ depends only on c_{aa} (the bottleneck angle β is 0 when the two inputs

are identical). We may then solve for c_{aa}. Substituting c_{aa} for $a \in \{1,2\}$ into Eq. (6.16) and noting that J_2 is strictly decreasing, we may solve for the bottleneck angle β from $q_{12}^{\times(\infty)}$ and thus obtain c_{12}, recovering C. □

Proof of Prop. 6.13. Since $\|x_1\| = \|x_2\|$, then $c_{11} = c_{22}$. Letting $c = c_{11} = c_{22}$, we have by Eq. (6.12) that

$$q_{ab}^{\times(D)} = \frac{\left(\frac{2}{\pi}J_2(\beta) - 1\right)}{15 + 2H\left(\frac{r_D}{c} + 1\right)^2}.$$

We will find a $\lambda > 0$ such that the limit L in Eq. (6.17) is finite and non-zero. Note that while evaluating the limit, we will drop (non-zero) constants of proportionality. Observing that both the numerator and denominator inside the limit L in Eq. (6.17) tend to 0 as $D \to \infty$ and using L'Hôpital's rule, we have

$$\begin{aligned}
L &= \lim_{D\to\infty} \frac{\frac{\mathrm{d}}{\mathrm{d}D} q_{ab}^{\times(D)} - \frac{\mathrm{d}}{\mathrm{d}D} q_{ab}^{\times(\infty)}}{\frac{\mathrm{d}}{\mathrm{d}D} e^{-\frac{D}{\lambda}}} \\
&= \lim_{D\to\infty} \frac{\frac{\mathrm{d}}{\mathrm{d}D} \frac{\left(\frac{2}{\pi}J_2(\beta)-1\right)}{15+2H\left(\frac{r_D}{c}+1\right)^2} - 0}{-\frac{1}{\lambda}e^{-\frac{D}{\lambda}}} \\
&\propto -\lim_{D\to\infty} e^{\frac{D}{\lambda}} \cdot \frac{\mathrm{d}}{\mathrm{d}D} \frac{1}{15 + 2H\left(\frac{r_D}{c} + 1\right)^2} \\
&= \lim_{D\to\infty} e^{\frac{D}{\lambda}} \cdot \frac{4H\left(\frac{r_D}{c} + 1\right)\frac{1}{c} \cdot \left(v_n + \frac{v_b}{1-v_w}\right) \cdot \frac{1}{v_w^D} \ln\left(\frac{1}{v_w}\right)}{\left[15 + 2H\left(\frac{r_D}{c} + 1\right)^2\right]^2} \\
&\propto \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D} \cdot \frac{\left(\frac{r_D}{c} + 1\right)}{\left[15 + 2H\left(\frac{r_D}{c} + 1\right)^2\right]^2}.
\end{aligned} \tag{31}$$

In the case $v_w > 1$, r_D tends to a finite positive limit as $D \to \infty$, so that the second fraction in the limit in Eq. (31) tends to a finite positive limit as well. We therefore have

$$L \propto \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D},$$

which is finite and non-zero (and exists) if and only if $\frac{e^{\frac{1}{\lambda}}}{v_w} = 1$, implying $\lambda = \ln(v_w)^{-1}$.

In the case $v_w < 1$, $r_D \to \infty$ as $D \to \infty$, so that Eq. (31) simplifies to

$$\begin{aligned} L &\propto \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D} \cdot \frac{\left(\frac{r_D}{c}\right)}{\left[2H\left(\frac{r_D}{c}\right)^2\right]^2} \\ &\propto \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D} \cdot \frac{1}{r_D^3} \\ &= \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D} \cdot \frac{1}{\left[\frac{v_n}{v_w^D} + \frac{v_b}{1-v_w}\left(\frac{1}{v_w^D} - 1\right)\right]^3} \\ &= \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D} \cdot \frac{1}{\left[\left(v_n + \frac{v_b}{1-v_w}\right)\frac{1}{v_w^D} - \frac{v_b}{1-v_w}\right]^3}. \end{aligned} \tag{32}$$

Since $\frac{1}{v_w^D} \to \infty$ as $D \to \infty$, then

$$L \propto \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{v_w^D} \cdot \frac{1}{\left[\left(v_n + \frac{v_b}{1-v_w}\right)\frac{1}{v_w^D}\right]^3} \propto \lim_{D\to\infty} v_w^{2D} e^{\frac{D}{\lambda}},$$

which is finite and non-zero (and exists) if and only if $v_w^2 e^{\frac{1}{\lambda}} = 1$, implying $\lambda = \ln\left(\frac{1}{v_w^2}\right)^{-1}$.

In the case $v_w = 1$, we again have $r_D \to \infty$ as $D \to \infty$ and thus still obtain Eq. (32). Substituting $v_w = 1$ and $r_D = v_n + v_b D$ into Eq. (32) gives

$$L \propto \lim_{D\to\infty} \frac{e^{\frac{D}{\lambda}}}{(v_n + v_b D)^3},$$

which is infinite for all finite $\lambda > 0$. □

Proof of Prop. 6.14. For part (a), let

$$\mathrm{d}\mu(h) = \prod_{m=1}^{H} \mathcal{N}(h_m; 0, C)\,\mathrm{d}h.$$

on $(\mathbb{R}^2)^H$. Note that μ is a non-degenerate normal distribution. Then we have

$$\begin{aligned} \mathrm{Cov}[F_1(x_a)^2, F_1(x_b)^2] &= \mathrm{E}[F_1(x_a)^2 F_2(x_b)^2] - \mathrm{E}[F_1(x_a)^2]\,\mathrm{E}[F_1(x_b)^2] \\ &= \int_{(\mathbb{R}^2)^H} \int_{(\mathbb{R}^2)^2} (y^a)^2 (y^b)^2 \mathcal{N}(y; 0, K)\,\mathrm{d}y\,\mathrm{d}\mu(h) - \mathrm{E}[F_1(x_a)^2]\,\mathrm{E}[F_1(x_b)^2] \\ &= \int_{(\mathbb{R}^2)^H} (2k_{ab}^2 + k_{aa}k_{bb})\,\mathrm{d}\mu(h) - \mathrm{E}[F_1(x_a)^2]\,\mathrm{E}[F_1(x_b)^2] \\ &= 2\int_{(\mathbb{R}^2)^H} k_{ab}^2\,\mathrm{d}\mu(h) + \int_{(\mathbb{R}^2)^H} k_{aa}k_{bb}\,\mathrm{d}\mu(h) - \mathrm{E}[F_1(x_a)^2]\,\mathrm{E}[F_1(x_b)^2]. \end{aligned}$$

Noting that $F_1(x_b)$ and $F_2(x_b)$ are identically distributed and recalling the proof of Prop. 6.10, we have

$$\begin{aligned}\operatorname{Cov}[F_1(x_a)^2, F_1(x_b)^2] &= 2\int_{(\mathbb{R}^2)^H} k_{ab}^2 \,\mathrm{d}\mu(h) + \int_{(\mathbb{R}^2)^H} k_{aa}k_{bb}\,\mathrm{d}\mu(h) - \operatorname{E}[F_1(x_a)^2]\operatorname{E}[F_2(x_b)^2] \\ &= 2\int_{(\mathbb{R}^2)^H} k_{ab}^2 \,\mathrm{d}\mu(h) + \operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2].\end{aligned}$$

Using again the fact that $F_1(x_b)$ and $F_2(x_b)$ are identically distributed, we have the correlation

$$\begin{aligned}q_{ab}^{(D)} &= \frac{\operatorname{Cov}[F_1(x_a)^2, F_1(x_b)^2]}{\sqrt{\operatorname{V}[F_1(x_a)^2]\operatorname{V}[F_1(x_b)^2]}} \\ &= 2\int_{(\mathbb{R}^2)^H} \frac{k_{ab}^2}{\sqrt{\operatorname{V}[F_1(x_a)^2]\operatorname{V}[F_1(x_b)^2]}}\,\mathrm{d}\mu(h) + \frac{\operatorname{Cov}[F_1(x_a)^2, F_2(x_b)^2]}{\sqrt{\operatorname{V}[F_1(x_a)^2]\operatorname{V}[F_1(x_b)^2]}} \\ &= 2\int_{(\mathbb{R}^2)^H} \frac{k_{ab}^2}{\sqrt{\operatorname{V}[F_1(x_a)^2]\operatorname{V}[F_1(x_b)^2]}}\,\mathrm{d}\mu(h) + q_{ab}^{\times(D)}.\end{aligned}$$

To make the dependence on the post-bottleneck depth D more explicit, we write

$$q_{ab}^{(D)} = 2\int_{(\mathbb{R}^2)^H} \frac{(k_{ab}^{(D)})^2}{\sqrt{\operatorname{V}[f_1^{(D)}(x_a)^2]\operatorname{V}[f_1^{(D)}(x_b)^2]}}\,\mathrm{d}\mu(h) + q_{ab}^{\times(D)}.$$

By Eq. (30), this becomes

$$q_{ab}^{(D)} = 2\int_{(\mathbb{R}^2)^H} \frac{(k_{ab}^{(D)})^2}{w_D^2 c_{aa}c_{bb}\sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}}+1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}}+1\right)^2\right)}}\,\mathrm{d}\mu(h) + q_{ab}^{\times(D)}.$$

The correlation at infinite depth is then

$$q_{ab}^{(\infty)} = \lim_{D\to\infty} 2\int_{(\mathbb{R}^2)^H} \frac{(k_{ab}^{(D)})^2}{w_D^2 c_{aa}c_{bb}\sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}}+1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}}+1\right)^2\right)}}\,\mathrm{d}\mu(h) + q_{ab}^{\times(\infty)}.$$

The limit can be moved inside the integral. To justify this, observe that the integrand (as a function of $h \in (\mathbb{R}^2)^H$) can be expressed as

$$
\begin{aligned}
I_D(h) &:= \frac{(k_{ab}^{(D)})^2}{w_D^2 c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&\leq \frac{k_{aa}^{(D)} k_{bb}^{(D)}}{w_D^2 c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&= \frac{\left(b_D + \frac{w_D}{H}\sum_i \phi(h_i^a)^2\right)\left(b_D + \frac{w_D}{H}\sum_i \phi(h_i^b)^2\right)}{w_D^2 c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&\leq \frac{\left(b_D + \frac{w_D}{H}\sum_i \phi(\max(h))^2\right)\left(b_D + \frac{w_D}{H}\sum_i \phi(\max(h))^2\right)}{w_D^2 c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&= \frac{\left(b_D + w_D \phi(\max(h))^2\right)^2}{w_D^2 c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&= \frac{\left(r_D + \phi(\max(h))^2\right)^2}{c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&\leq \frac{2 r_D^2}{c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}} \\
&+ \frac{2 \phi(\max(h))^4}{c_{aa} c_{bb} \sqrt{\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{aa}} + 1\right)^2\right)\left(\frac{15}{H} + 2\left(\frac{r_D}{c_{bb}} + 1\right)^2\right)}}.
\end{aligned}
$$

Recall that $r_D \to \frac{v_b}{v_w - 1}$ if $v_w > 1$ and $r_D \to \infty$ otherwise. In either case, it is easy to verify that the first term and the denominator of the second term converge to non-negative numbers independent of D. Therefore, there exist positive constants A and B such that

$$I_D(h) < A + B\phi(\max(h))^4 \text{ for sufficiently large } D.$$

Note the right-hand side is integrable with respect to the non-degenerate Gaussian measure μ since it is a piecewise polynomial in h (with finitely many pieces). We can therefore use the Dominated Convergence Theorem. Recall also that the NNGP kernel (post-bottleneck) degenerates to a constant-element kernel corresponding to a correlation matrix of 1's given

any fixed input h from the bottleneck layer. Using the Dominated Convergence Theorem twice, we therefore have

$$
\begin{aligned}
q_{ab}^{(\infty)} &= 2\int_{(\mathbb{R}^2)^H} \lim_{D\to\infty} \frac{(k_{ab}^{(D)})^2}{w_D^2 c_{aa} c_{bb} \prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{H} + 2\left(\frac{r_D}{c}+1\right)^2}} \,\mathrm{d}\mu(h) + q_{ab}^{\times(\infty)} \\
&= 2\int_{(\mathbb{R}^2)^H} \lim_{D\to\infty} \frac{(k_{ab}^{(D)})^2}{k_{aa}^{(D)} k_{bb}^{(D)}} \frac{k_{aa}^{(D)} k_{bb}^{(D)}}{w_D^2 c_{aa} c_{bb} \prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{H} + 2\left(\frac{r_D}{c}+1\right)^2}} \,\mathrm{d}\mu(h) + q_{ab}^{\times(\infty)} \\
&= 2\int_{(\mathbb{R}^2)^H} \lim_{D\to\infty} \frac{k_{aa}^{(D)} k_{bb}^{(D)}}{w_D^2 c_{aa} c_{bb} \prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{H} + 2\left(\frac{r_D}{c}+1\right)^2}} \,\mathrm{d}\mu(h) + q_{ab}^{\times(\infty)} \\
&= \lim_{D\to\infty} \frac{2}{w_D^2 c_{aa} c_{bb} \prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{H} + 2\left(\frac{r_D}{c}+1\right)^2}} \int_{(\mathbb{R}^2)^H} k_{aa}^{(D)} k_{bb}^{(D)} \,\mathrm{d}\mu(h) + q_{ab}^{\times(\infty)} \\
&= \lim_{D\to\infty} \frac{2}{w_D^2 c_{aa} c_{bb} \prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{H} + 2\left(\frac{r_D}{c}+1\right)^2}} \\
&\cdot \left((b_D + w_D c_{aa})(b_D + w_D c_{bb}) + \mathrm{Cov}[f_1^{(D)}(x_a)^2, f_2^{(D)}(x_b)^2]\right) + q_{ab}^{\times(\infty)} \\
&= \lim_{D\to\infty} \frac{2(b_D + w_D c_{aa})(b_D + w_D c_{bb})}{w_D^2 c_{aa} c_{bb} \prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{H} + 2\left(\frac{r_D}{c}+1\right)^2}} + 3q_{ab}^{\times(\infty)} \\
&= \lim_{D\to\infty} \frac{\left(\frac{r_D}{c_{aa}}+1\right)\left(\frac{r_D}{c_{bb}}\right)}{\prod_{c=c_{aa},c_{bb}} \sqrt{\frac{15}{2H} + \left(\frac{r_D}{c}+1\right)^2}} + 3q_{ab}^{\times(\infty)}.
\end{aligned}
$$

Evaluating the limit by recalling the limit of r_D, we obtain Eq. (6.19) as desired.

For part (b), given $(Q^{(\infty)}, \mathrm{diag}(G))$, we will show that we recover G. We can obtain $\mathrm{diag}(C)$ from $\mathrm{diag}(G)$ by applying the NNGP kernel propagation defined in Eq. (6.9). Given $\mathrm{diag}(C)$, we can solve for $Q^{\times(\infty)}$ using Eq. (6.19) for the case $v_w > 1$ in Prop. 6.14. We can then obtain G by Prop. 6.12 (b). □

I.1 Other nonlinearities

The following is the proof for the proposition linking the sinusoidal nonlinearity in Eq. (6.20) to the RBF kernel.

Proof of Prop. 6.15. First we prove part (a). The case $\mu = 0$ is trivial. So, consider $\mu \geq 1$. Define the function $F_\phi : \mathbb{R}^3 \mapsto \mathbb{R}$ by

$$F_\phi(a, b, c) = \int_{\mathbb{R}^2} \phi(z_1)\phi(z_2)\mathcal{N}\left(z; 0, \begin{bmatrix} a & b \\ b & c \end{bmatrix}\right) \mathrm{d}z, \tag{33}$$

where ϕ is the sinusoidal nonlinearity in Eq. (6.20) and

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}.$$

The NNGP kernel recursion can then be written as

$$K^{(\mu+1)}(x, x') = v_b + v_w F_\phi[K^{(\mu)}(x, x), K^{(\mu)}(x, x'), K^{(\mu)}(x', x')].$$

All we need to show is that

$$F_\phi(a, b, c) = e^{-\frac{1}{2}(a+c-2b)}. \tag{34}$$

Let X be a 2×2 matrix with columns $x_1, x_2 \in \mathbb{R}^2$ such that

$$X^\top X = \begin{bmatrix} a & b \\ b & c \end{bmatrix}. \tag{35}$$

Such a matrix X exists since the matrix on the right side is symmetric positive semidefinite. Performing the change of variables $z = X^\top w$, Eq. (33) becomes

$$\begin{aligned} F_{phi}(a, b, c) &= \int_{\mathbb{R}^2} \phi(x_1^\top w)\phi(x_2^\top w)\mathcal{N}(w; 0, I)\, \mathrm{d}w \\ &= \frac{1}{2\pi} \int_{\mathbb{R}^2} \phi(w \cdot x_1)\phi(w \cdot x_2) e^{-\frac{1}{2}\|w\|^2}\, \mathrm{d}w. \end{aligned}$$

The product of activations in the integrand can be rewritten as

$$\begin{aligned} \phi(w \cdot x_1)\phi(w \cdot x_2) &= [\cos(w \cdot x_1) + \sin(w \cdot x_1)][\cos(w \cdot x_2) + \sin(w \cdot x_2)] \\ &= \cos(w \cdot x_1)\cos(w \cdot x_2) + \sin(w \cdot x_1)\sin(w \cdot x_2) \\ &+ \cos(w \cdot x_1)\sin(w \cdot x_2) + \sin(w \cdot x_1)\cos(w \cdot x_2) \\ &= \cos[w \cdot (x_1 - x_2)] + \sin[w \cdot (x_1 + x_2)]. \end{aligned}$$

We therefore have

$$\begin{aligned} F_\phi(a, b, c) &= \frac{1}{2\pi} \int_{\mathbb{R}^2} \cos[w \cdot (x_1 - x_2)] e^{-\frac{1}{2}\|w\|^2}\, \mathrm{d}w \\ &+ \frac{1}{2\pi} \int_{\mathbb{R}^2} \sin[w \cdot (x_1 + x_2)] e^{-\frac{1}{2}\|w\|^2}\, \mathrm{d}w. \end{aligned}$$

The integrand of the second integral on the right side is odd in w for all x_1 and x_2, and thus this integral is zero. We can therefore replace $x_1 + x_2$ with $x_1 - x_2$ in the second integral and

multiply the integral by the imaginary unit i without changing its value:

$$\begin{aligned}
F_\phi(a,b,c) &= \frac{1}{2\pi}\int_{\mathbb{R}^2} \cos[w\cdot(x_1-x_2)]e^{-\frac{1}{2}\|w\|^2}\,\mathrm{d}w \\
&+ i\frac{1}{2\pi}\int_{\mathbb{R}^2} \sin[w\cdot(x_1-x_2)]e^{-\frac{1}{2}\|w\|^2}\,\mathrm{d}w \\
&= \frac{1}{2\pi}\int_{\mathbb{R}^2} \left(\cos[w\cdot(x_1-x_2)] + i\sin[w\cdot(x_1-x_2)]\right)e^{-\frac{1}{2}\|w\|^2}\,\mathrm{d}w \\
&= \frac{1}{2\pi}\int_{\mathbb{R}^2} e^{iw\cdot(x_1-x_2)}e^{-\frac{1}{2}\|w\|^2}\,\mathrm{d}w \\
&= e^{-\frac{1}{2}\|x_1-x_2\|^2},
\end{aligned}$$

where the last line holds because the Gaussian is an eigenfunction of the Fourier transform. Using Eq. (35), this becomes

$$\begin{aligned}
F_\phi(a,b,c) &= e^{-\frac{1}{2}(\|x_1\|^2+\|x_2\|^2-2x_1\cdot x_2)} \\
&= e^{-\frac{1}{2}(a+c-2b)},
\end{aligned}$$

giving us Eq. (34) as desired.

For part (b), consider two distinct inputs $x, x' \in \mathbb{R}^M$ and define

$$v^{(\mu)} = K^{(\mu)}(x,x') \text{ and } c^{(\mu)} = \frac{K^{(\mu)}(x,x')}{\sqrt{K^{(\mu)}(x,x)K^{(\mu)}(x',x')}}.$$

All we need to show is that $(v^{(\mu)}, c^{(\mu)})$ has a globally attractive fixed point (v_*, c_*) of the form given in the statement of the proposition. The dynamics of $v^{(\mu)}$ is given by $v^{(\mu+1)} = f_v(v^{(\mu)})$ where $f_v : (0,\infty) \mapsto (0,\infty)$ is given by

$$\begin{aligned}
f_v(v) &= v_b + v_w F_\phi(v,v,v) \\
&= v_b + v_w.
\end{aligned}$$

It is thus trivial that the global fixed point of f_v is $v_*(v_b, v_w) = v_b + v_w$. The dynamics of $c^{(\mu)}$ is given by $c^{(\mu+1)} = f_c(c^{(\mu)})$ where $f_c : [-1,1] \mapsto [-1,1]$ is given by

$$f_c(c) = v_b + v_w F_\phi(v_*, v_* c, v_*),$$

where we approximate $v^{(\mu)}$ with v_*; this approximation becomes exact in the deep limit. Substituting in $v_* = v_b + v_w$, we get

$$f_c(c) = \frac{v_b}{v_b+v_w} + \frac{v_w}{v_b+v_w}e^{(v_b+v_w)(c-1)}.$$

We make use of the fact that if a function $f : \mathbb{R} \mapsto \mathbb{R}$ has strictly positive first and second derivatives, then f has either no fixed points, one fixed point that is neither stable nor unstable, or two fixed points $c_1 < c_2$ where c_1 is stable with basin of attraction $(-\infty, c_2)$ and c_2 is unstable. Observe that $f_c(1) = 1$ and $f_c'(1) = v_w$. If $v_w < 1$, then $c = 1$ is a stable fixed

point and is thus globally attractive on $[-1, 1]$, thus establishing $c_*(v_b, v_w) = 1$ for $v_w < 1$. On the other hand, if $v_w > 1$, then $c = 1$ is an unstable fixed point. Since $f_c(0) > 0$ and $f_c(1) = 1$, then by the Intermediate Value Theorem, f_c has a stable fixed point $c' \in (0, 1)$ that is globally attractive on $[-1, 1)$. Given the initial condition $c^{(0)} = \frac{x^\top x'}{\|x\|\|x'\|} < 1$ (since $x \neq x'$) and that $c = 1$ is an unstable fixed point, then we must have $c_*(v_b, v_w) < 1$ for $v_w > 1$ and thus in particular $c_*(v_b, v_w) = c'$, concluding the proof. □

Proof of Prop. 6.17. Let H be the bottleneck width, and let $h_1, h_2 \in \mathbb{R}^H$ be the bottleneck preactivations—i.e., the outputs of the pre-bottleneck NNGP component—given network inputs $x_1, x_2 \in \mathbb{R}^M$. Let μ be the (Gaussian) probability measure associated to (h_1, h_2). Let $z_1, z_2 \in \mathbb{R}^H$ be the corresponding bottleneck activations that are fed into the post-bottleneck NNGP component, where $z_a = \frac{1}{\sqrt{H}}\phi(h_a)$ for $a = 1, 2$. Finally, let $K^{(D)} : \mathbb{R}^H \times \mathbb{R}^H \mapsto \mathbb{R}$ be the kernel of the post-bottleneck NNGP component assuming post-bottleneck depth D. Then the PDF of the bottleneck NNGP outputs is given by

$$p^{(D)}(y) = \int_{(\mathbb{R}^H)^2} \mathcal{N}(y; 0, K^{(D)}(Z, Z) + v_n I) \, \mathrm{d}\mu(h_1, h_2),$$

where $K^{(D)}(Z, Z)$ is a 2×2 matrix with entries $K^{(D)}(z_a, z_b)$ for $a, b = 1, 2$. Using the fact that for any 2×2 positive semidefinite matrix A with eigenvalues $\lambda_1, \lambda_2 \geq 0$,

$$\det(A + v_n I) = (\lambda_1 + v_n)(\lambda_2 + v_n) \geq v_n^2,$$

we have the bound

$$\begin{aligned}
\mathcal{N}(y; 0, K^{(D)}(Z, Z) + v_n I) &\leq \frac{1}{2\pi} \det[K^{(D)}(Z, Z) + v_n I]^{-\frac{1}{2}} \\
&\leq \frac{1}{2\pi}(v_n^2)^{-\frac{1}{2}} \\
&= \frac{v_n}{2\pi}.
\end{aligned}$$

Since the bound is clearly an integrable function with respect to μ and since the bound holds for all D, then we may apply the Bounded Convergence Theorem. By the continuity of the matrix determinant and inversion operations, the PDF converges to

$$\begin{aligned}
p^{(\infty)}(y) &= \lim_{D \to \infty} p^{(D)}(y) \\
&= \int_{(\mathbb{R}^H)^2} \mathcal{N}(y; 0, \lim_{D \to \infty} K^{(D)}(Z, Z) + v_n I) \, \mathrm{d}\mu(h_1, h_2) \\
&= \int_{(\mathbb{R}^H)^2} \mathcal{N}(y; 0, K^{(\infty)}(Z, Z) + v_n I) \, \mathrm{d}\mu(h_1, h_2). \qquad (36)
\end{aligned}$$

Define the set

$$\begin{aligned}
S &= \left\{(h_1, h_2) \in (\mathbb{R}^H)^2 : z_1 = z_2\right\} \\
&= \left\{(h_1, h_2) \in (\mathbb{R}^H)^2 : \frac{1}{\sqrt{H}}\phi(h_1) = \frac{1}{\sqrt{H}}\phi(h_2)\right\} \\
&= \{(h_1, h_2) \in (\mathbb{R}^H)^2 : \phi(h_1) = \phi(h_2)\} \\
&= \{(h_1, h_2) \in (\mathbb{R}^H)^2 : \exists n \in \mathbb{Z}^H \mid h_1 = h_2 + 2n\pi\} \\
&= \bigcup_{n \in \mathbb{Z}^H} \{(h, h + 2n\pi) : h \in \mathbb{R}^H\}.
\end{aligned}$$

We see that S is a countable disjoint union of H-dimensional planes embedded in a $2h$-dimensional space. Since the network inputs x_1 and x_2 are distinct, then μ is a non-degenerate Gaussian distribution on $(\mathbb{R}^H)^2$ so that $\mu(S) = 0$. We can therefore remove S from the region of integration in Eq. (36), so that z_1 and z_2 are distinct inputs into $K^{(\infty)}$ under the integral. We thus evaluate the covariance of the integrand in Eq. (36) using Prop. 6.15 (b) and Eq. (6.23) and get

$$\begin{aligned}
p^{(\infty)}(y) &= \int_{(\mathbb{R}^H)^2 \setminus S} \mathcal{N}(y; 0, K^{(\infty)}(Z, Z) + v_n I)\, \mathrm{d}\mu(h_1, h_2) \\
&= \int_{(\mathbb{R}^H)^2 \setminus S} \mathcal{N}\left(y; 0, v_*(v_b, v_w) \begin{bmatrix} 1 & c_*(v_b, v_w) \\ c_*(v_b, v_w) & 1 \end{bmatrix} + v_n I\right) \mathrm{d}\mu(h_1, h_2) \\
&= \mathcal{N}\left(y; 0, v_*(v_b, v_w) \begin{bmatrix} 1 & c_*(v_b, v_w) \\ c_*(v_b, v_w) & 1 \end{bmatrix} + v_n I\right),
\end{aligned}$$

which according to Prop. 6.15 (b) is precisely the deep limit with no bottleneck given two distinct inputs, thus establishing Eq. (6.26). □

Vita

Devanshu Agrawal is a PhD student in the Data Science and Engineering program in the Bredesen Center at the University of Tennessee, Knoxville and works as a research assistant in the Computational Sciences and Engineering division at Oak Ridge National Laboratory. Before joining the Bredesen Center, he spent a while at East Tennessee State University, where he received bachelors degrees in mathematics and physics, a masters degree in mathematics, and where he taught calculus after that. Devanshu is primarily interested in Bayesian deep learning and in both its practical and foundational implications.