Combining Bayesian and Deep Learning Methods in Computer Vision Problems

A Dissertation

submitted to the designated
by the General Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Panagiotis Dimitrakopoulos

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

University of Ioannina

July 2024

Advisory Committee:

- **Christophoros Nikou**, Professor , Department of Computer Science and Engineering, University of Ioannina
- **Giorgos Sfikas**, Assistant Professor, Department of Surveying and Geoinformatics Engineering, University of West Attica
- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina

Examining Committee:

- **Christophoros Nikou**, Professor , Department of Computer Science and Engineering, University of Ioannina
- **Giorgos Sfikas**, Assistant Professor, Department of Surveying and Geoinformatics Engineering, University of West Attica
- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Lisimachos P. Kondi**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Konstantinos D. Blekas** , Professor, Department of Computer Science and Engineering, University of Ioannina
- **Stefanos Zafeiriou**, Professor, Department of Computing, Imperial College London
- **Nikos Paragios**, Professor, School of Engineering (CentraleSupelec), University of Paris - Saclay

Dedication

To all the people who supported me on this incredible journey!

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Christoforos Nikou, for granting me the opportunity to pursue this PhD. His guidance and support throughout these years have been invaluable. Particularly valuable was the intellectual and research freedom he offered, allowing me to explore my research topic and direction without restrictions. Furthermore, his financial support through the "BESSARION" research project position was instrumental in making this dissertation possible. I would also like to extend my deepest thanks to my co-advisor, Prof. Giorgos Sfikas. His contributions were crucial to this thesis. Throughout my PhD year whenever I brought him an idea with some preliminary experiments, he helped me to grasp the potential and significance of it and how we can further improve it. He instilled in me the importance of asking the right scientific questions which really shape my scientific way of thinking. His unwavering support and constant presence were invaluable. Moreover, his belief in our work, exceeding even my own, and his guidance in fostering high aspirations and dreams for the future were truly inspiring.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Panagiotis Dimitrakopoulos, Ph.D., Department of Computer Science and Engineering, University of Ioannina, Greece, July 2024.
Combining Bayesian and Deep Learning Methods in Computer Vision Problems.
Advisor: Christophoros Nikou, Professor.

Neural networks dominate computer vision tasks, yet their predictions often lack reliability. Bayesian Deep Learning (BDL) offers a solution by treating model parameters as random variables. This approach leads to well-calibrated predictions and handles distribution shifts better than deterministic methods. However, computational limitations prevents its widespread applicability. This thesis focuses on developing efficient BDL methods for high-dimensional parameter spaces, which are applied on various computer vision tasks, including image classification, segmentation, and object detection. Specifically we propose lightweight Bayesian modules for robust and probabilistic object detection via efficient stochastic feature fusion. Additionally, we introduce a novel hypernetwork-based method for incorporating Bayesian inference to large vision models. Finally we experimented with a structured posterior distribution, which efficiently captures correlations between weights, leading to improved calibration and uncertainty quantification. This research paves the way for the development of more interpretable and reliable machine learning models.

Εκτεταμενη Περιληψη

Παναγιώτης Δημητρακόπουλος, Δ.Δ., Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2024.
Συνδυασμός Μπεϋζιανών Μεθόδων και Βαθιάς Μάθησης για Προβλήματα Υπολογιστικής Όρασης.
Επιβλέπων: Χριστόφορος Νίκου, Καθηγητής.

Τα νευρωνικά δίκτυα έχουν αναδειχθεί σε κυρίαρχη επιλογή για την μηχανική μάθηση τα τελευταία χρόνια, κυριαρχώντας σε πλήθος επιστημονικών πεδίων. Η άνοδός τους οφείλεται σε μεγάλο βαθμό στην ανάπτυξη των συνελικτών νευρωνικών δικτύων, ιδιαίτερα στον τομέα της υπολογιστικής όρασης. Παρά την αδιαμφισβήτητη επιτυχία και τα εξαιρετικά αποτελέσματά τους, τα κλασικά νευρωνικά δίκτυα παρουσιάζουν και ορισμένα μειονεκτήματα. Ένα από τα πιο σημαντικά είναι η πολύ μεγαλή βεβαιότητα στις προβλέψεις τους (ειδικά τις λανθασμένες), φαινόμενο που γίνεται πιο έντονο όταν το μοντέλο καλείται να προβλέψει δεδομένα που διαφέρουν από τα δεδομένα εκπαίδευσης.

Τα Μπεϋζιανά νευρωνικά δίκτυα έρχονται να προσφέρουν μια λύση σε αυτά τα προβλήματα, καθώς συνδυάζουν την αποτελεσματικότητα της βαθιάς μάθησης με τα πλεονεκτήματα της πιθανοτικής προσέγγισης που βασίζεται στην Μπεϋζιανή ανάλυση. Κατά την εκπαίδευση Μπεϋζιανών νευρωνικών δικτύων, η εστίαση τίθεται στην εκτίμηση της εκ των υστέρων κατανομής των παραμέτρων, δεδομένου του συνόλου εκπαίδευσης. Η Μπεϋζιανή αυτή προσέγγιση της μάθησης προσφέρει πλήθος πλεονεκτημάτων, όπως είναι οι αξιόπιστες προβλέψεις στα δεδομένα ελέγχου, που συνοδεύονται από εκτιμίσεις αβεβαιότητας. Βελτιωμένες προβλέψεις σε δεδομένα που διαφέρουν από το σύνολο εκπαίδευσης. Καθώς επίσης και ένα πλαίσιο για την εκμάθηση των υπερπαραμέτρων. Συνολικά, τα Μπεϋζιανά νευρωνικά δίκτυα αποτελούν μια ελπιδοφόρα εναλλακτική λύση στα κλασικά νευρωνικά δίκτυα, προσφέροντας πιο αξιόπιστες και γενικεύσιμες προβλέψεις.

Παρά τα πλεονεκτήματά τους, τα Μπεϋζιανά νευρωνικά δίκτυα φέρουν και μειονεκτήματα. Ένα από τα πιο σημαντικά είναι η αυξημένη υπολογιστική ισχύς που απαιτούν σε σύγκριση με τα κλασικά νευρωνικά δίκτυα. Καθώς ο αριθμός των παραμέτρων στα νευρωνικά δίκτυα αυξάνεται με την πάροδο του χρόνου, το ενδιαφέρον στρέφεται σε πιθανοτικές μεθόδους και τεχνικές που συνδυάζουν αποδοτικότητα με χαμηλή κατανάλωση υπολογιστικών πόρων. Στόχος λοιπόν είναι η αποδοτική πιθανοτική μοντελοποίηση παραμέτρων που οδηγεί σε εύρωστες προβλέψεις και αξιόπιστες τιμές αβεβαιότητας στα αποτελέσματα. Η παρούσα διδακτορική διατριβή εστιάζει στην ανάπτυξη αποδοτικών Μπεϋζιανών μοντέλων και τεχνικών με εφαρμογή σε προβλήματα υπολογιστικής όρασης, όπως η κατηγοριοποίηση εικόνων, η ανίχνευση αντικειμένων και σημασιολογική κατάτμηση.

Ειδικότερα, η διατριβή εστιάζει στο πρόβλημα της αξιόπιστης και πιθανοτικής ανίχνευσης αντικειμένων σε εικόνες. Προτείνεται μια νέα Μπεϋζιανή τεχνική για την εξαγωγή χαρακτηριστικών, με στόχο τη δημιουργία ευρύτερων περιγραφών εικόνων που μπορούν να αξιοποιηθούν για βελτιωμένα αποτελέσματα. Η τεχνική αυτή βασίζεται στο υποδικτύα Feature Pyramid Networks [1], τα οποία αντλούν χαρακτηριστικά από διαφορετικά επίπεδα ενός νευρωνικού δικτύου και τα επεξεργάζονται με βάση μια αρχιτεκτονική πυραμιδοειδούς σχήματος, ώστε να γίνουν πιο περιγραφικά. Χαρακτηριστικό του προτεινόμενου υποδικτύου είναι η στοχαστική φύση τής αρχιτεκτονικής του, ή οποία μαθαίνεται από τα δεδομένα εκπαίδευσης με τη χρήση Variational Inference, Με το τρόπο αυτό η αρχιτεκτονική του μπορέι και προσαρμόζεται ανάλογα με τα ιδιαίτερα χαρακτηριστικά του συνόλου εκπαίδευσης, προσφέροντας εξατομικευμένα χαρακτηριστικά σαν έξοδο. Επιπλέον, η πιθανοτική προσέγγιση ενσωματώνει στοχαστικότητα στο μοντέλο ανίχνευσης, επιτρέποντας την παροχή πιθανοτικών προβλέψεων με εκτίμιση αβεβαιότητας στις τελικές προβλέψεις ανίχνευσης. Το προτεινόμενο μοντέλο αξιολογήθηκε σε συνδυασμό με σύγχρονα μοντέλα ανίχνευσης, επιτυγχάνοντας αξιόπιστα αποτελέσματα σε διάφορα σύνολα δεδομένων.

Στη συνέχεια, η διατριβή εστιάζει στην εφαρμογή της Μπεϋζιανής ανάλυσης σε νευρωνικά δίκτυα με παραμετρους υψηλής διαστασης, προτείνοντας τεχνικές που υπερβαίνουν αυτό το πρόβλημα ορίζοντας έναν εναλλακτικό χαμηλής διαστασης χώρο στον οποίο εφαρμόζεται η πιθανολογική ανάλυση. Συγκεκριμένα, προτείνεται ένα υπερδίκτυο με λίγες παραμέτρους που λειτουργεί παράλληλα με το κύριο δίκτυο. Σκοπός του υπερδικτύου είναι η παραγωγή στοχαστικών παραμέτρων που

εισάγουν τυχαιότητα στο κύριο δίκτυο. Η Μπεϋζιανή ανάλυση πραγματοποιείται μόνο στις χαμηλών διαστάσεων παραμέτρους του υπερδικτύου, επιτρέποντας την υιοθέτηση πιθανοτικών μεθόδων μεγάλης ευελιξίας που θα ήταν αδύνατες σε υψηλής διάστασης χώρους (όπως ο χώρος παραμέτρων του κύριου δικτύου). Ο χωρος διαστάσης του υπερδικτύου διατηρείται μικρός μέσω της χρήσης Implicit Neural Representation [2]. Τα οφέλη αυτής της έμμεσης πιθανοστατιστικής προσέγγισης αξιολογήθηκαν σε μια σειρά πειραμάτων.

Η διατριβή ολοκληρώνεται με την πρόταση μιας νέας μεθόδου για την αποτελεσματική μοντελοποίηση των συσχετίσεων μεταξύ των βαρών ενός νευρωνικού δικτύου. Η προτεινόμενη μέθοδος παρουσιάζει σημαντικά μειωμένο υπολογιστικό κόστος σε σύγκριση με εναλλακτικές προσεγγίσεις. Συγκεκριμένα, στο πλαίσιο του Variational Inference, όπου η πραγματική εκ των υστέρων κατανομή των βαρών προσεγγίζεται από μια πολυδιάστατη Γκαουσιανή κατανομή, προτείνεται μια παραμετροποίηση του πίνακα συνδιακύμανσης βασισμένη στη θεωρία των κυκλωτικών πινάκων. Η παραμετροποίηση αυτή δίνει έμφαση στη μοντελοποίηση συγκεκριμένων συσχετίσεων μεταξύ των παραμέτρων του νευρωνικού δικτύου. Η πειραματική αξιολόγηση της παραμετροποίησης επιβεβαιώνει τα οφέλη της τόσο σαν εκ των υστέρων όσο και σαν εκ των προτέρων κατανομής, προσφέροντας μια πιο αποδοτική επιλογή με αρκετά καλά αποτελέσματα.

Η διατριβή αυτή υποστηρίζει ότι οι Μπεϋζιανές μέθοδοι αποτελούν όχι μόνο μια αξιόπιστη εναλλακτική λύση εκπαίδευσης για σύγχρονα νευρωνικά δίκτυα με πολλές παραμέτρους, αλλά και μια αποδοτική λύση, η οποία μπορεί να έχει χαμηλό υπολογιστικό κόστος. Η ανάπτυξη πιθανοτικών μεθόδων που παρέχουν αξιόπιστες εκτιμήσεις αβεβαιότητας στις προβλέψεις τους μπορεί να αποτελέσει λύση προς την κατεύθυνση μοντέλων με πιο ενημερωτικές προβλέψεις, κατάλληλων για εφαρμογές υψηλής κρισιμότητας, όπως ιατρικές και εφαρμογές αυτόνομης οδήγησης.

Chapter 1

Introduction

Neural networks (NNs) are the cornerstone of many fields where machine learning models are applicable. While shallow networks, primarily Multi-Layer Perceptrons, have limited descriptive power, the ability to increase the number of hidden units has paved the way for deeper architectures like deep NNs and the emergence of deep learning. In recent years, deep learning models have witnessed an exponential growth in popularity, not only within research but also in everyday life. This widespread adoption is further fueled by advancements with the huge popularity of large foundational models.

Despite their massive success and widespread deployment, NNs have several critical shortcomings. Their predictions are often characterized by overconfidence, which becomes especially problematic when there is a significant distribution shift between the training and test data. Ideally, we would like the network to identify cases where the test data point is statistically far from the training set distribution and produce a quantifiable measure of uncertainty. This becomes crucial for applications like autonomous driving or medical diagnosis, where making confident decisions on potentially out-of-distribution data is critical. Even in cases where outputs are probabilities (e.g., softmax outputs), standard NNs often suffer from poorly calibrated probabil-

ity scores. Furthermore, neural networks are known to be vulnerable to distribution shifts. Additionally, the process of selecting hyperparameters through cross-validation or meta-learning can be challenging due to the inherent difficulties associated with these approaches.

Bayesian Deep Learning (BDL) emerges as a solution to the aforementioned pathologies of traditional deep learning models. It aims to combine the effectiveness of deep learning with the advantages of Bayesian inference. From a Bayesian perspective, model parameters are treated as probabilistic random variables. Instead of training the network in the traditional sense, BDL focuses on computing the posterior distribution of these parameters given the observed training data. This Bayesian approach offers several benefits, such as well-calibrated predictive distributions, a framework for hyperparameter selection, and improved handling of issues like training-test distribution shift and catastrophic forgetting. Despite its advantages, Bayesian inference also has limitations. In most practical models, including those outside the realm of deep learning, it often requires approximate inference techniques. However, even these approximations can introduce significant computational overhead.

In this thesis, we leverage this paradigm to develop new methods that unify the complementary benefits of deep learning and probabilistic modeling. Deep learning excels at extracting high-dimensional feature representations from data (e.g., images). Probabilistic modeling, on the other hand, allows for flexible reasoning about the distribution over the model's parameters. We demonstrate how this combination address the aforementioned challenges and provide robust models in computer vision applications.

1.1 Efficient Bayesian Deep Learning in Computer Vision

As neural networks grow increasingly complex, with parameter size scaling into the billions, straightforward Bayesian modeling becomes computationally infeasible. This naturally leads to a focus on developing and applying approximate Bayesian inference techniques for handling these extremely high-dimensional parameter spaces. The goal is to achieve reliable uncertainty quantification, allowing for efficient real-time predictions, a crucial aspect of modern machine learning applications.

This thesis focuses on modeling lightweight Bayesian modules and techniques

applicable to various computer vision problems, particularly image classification, semantic segmentation, and object detection. These techniques offer advantages in terms of interpretability and uncertainty quantification compared to traditional deep learning approaches. In this section, we describe the application domains of Bayesian deep learning considered in this thesis and outline our contributions to address each problem.

We first study the problem of probabilistic object detection. Our work focuses on developing lightweight Bayesian techniques for effective feature fusion, which aims to create robust and stochastic deep representations of real-world images suitable for various downstream tasks (i.e. segmentation detection). We then steer our focus to Low-Dimensional Deep Learning, where we discuss its challenges. We propose novel techniques for applying Bayesian treatment of deep learning while maintaining high accuracy. Then we conclude with weight correlation modeling. This approach seeks to learn a structured posterior distribution that efficiently captures the inherent correlations between neural network weights. Our work in this area contributes to improved model interpretability and uncertainty quantification.

1.1.1 Probabilistic Object Detection

Motivation & Problem Statement

Object detection and instance segmentation are two fundamental problems in computer vision, posing significant challenges. These challenges arise from the need to detect or segment multiple objects at various scales, locations, and under diverse conditions. While deep convolutional neural networks (CNNs) have shown remarkable progress in object localization and segmentation in recent years, their ability to perform safely in diverse environments remains an area of active research. Furthermore, quantifying both epistemic (knowledge-based) and aleatoric (inherent data noise) uncertainty is crucial for safe decision-making in safety-critical applications. In such applications, the reliability of an agent's perception and decision-making is paramount [3] .

Contributions

Many computer vision methods using neural networks rely directly on features extracted from the original image data. These features are typically obtained using a

neural network (often referred to as a backbone) due to their ability to generate compact features rich in semantic information [4]. Feature fusion techniques introduce an additional sub-network that efficiently combines the extracted feature information from an input image. This aims to generate even richer and more compact features, ultimately leading to improved prediction performance in object detection tasks. Our work specifically focused on the feature fusion process to achieve robust and probabilistic object detection models.

Following previous works [5] we build an efficient feature fusion network. This network adopts elements from state-of-the-art fusion networks and initialize a complex fusion architecture, which subsequently is pruned to its more efficient counterpart, reducing the excess parameters and computational cost. We opt to learn to highlight the network components that are more suitable to the specific task and dataset that it is trained on. The process of pruning the network is formulated as learning the posterior distribution over a set of architectural weight variables given the training dataset. By treating even this small subset of parameters probabilistically we are able to introduce stochasticity to the main network [6] and acquire uncertainty in the final predictions.

Our experiments show that the models produced using the proposed method, combined with various object detectors and segmentation networks, produce state-of-the-art results. Furthermore, we utilized a method for numerically quantifying uncertainty estimates in object detection. To this end we utilize a non-maximum-suppression algorithm [7], which takes into account uncertainties in bounding box locations. We directly link the quality of uncertainty estimates with the numerical benefits gained from using this method and leverage this connection to evaluate our approach.

1.1.2 Low Dimensional Bayesian Deep Learning

Motivation & Problem Statement

While current deep learning models can accommodate millions of trainable parameters, performing exact Bayesian inference on such complex, high-dimensional settings is infeasible. Applying Bayes' rule directly to derive the posterior over parameter weights becomes impractical. Consequently, practitioners rely on approximate inference methods, often introducing specific restrictions or assumptions. However, even

with these simplifications, existing approximate inference schemes might not be able to efficiently handle millions of parameters, necessitating even more stringent limitations on the approximate posterior distribution.

Low-dimensional Bayesian deep learning emerges as a solution to this very challenge. The goal is to find an approximation where probabilistic reasoning over the full network parameters is cast to approximate inference in a smaller dimensional space. Ideally we would like this approximation to minimize the discrepancy to approximate posterior over the full parameters space. Several solutions have been proposed in this respect, rehashing and adapting older methods [8] [9] or putting forward completely fresh approaches [10].

Contributions

We propose a novel framework modeling for applying flexible low dimensional inference in deep learning networks. In our method we augment each weight of the main network with a multiplicative nuisance factor. These factors are in turn obtained by evaluating a flexible hypernetwork designed to be compact and reusable across the main network. We define a prior over the nuisance factors implicitly by probabilistically modeling the hypernetwork parameters. This prior combined with a Dirac distribution over the main network weights results in a Bayesian treatment of the main network. We leverage the power of implicit neural representations (INRs), inspired by their success in various fields [2]. These networks and their data modeling procedure unable as to keep the hypernetwork parameters to a very small scale. Finally the low dimensionality of the hypernetwork parameter space allows as to use any off-the-shelf multi-modal approximate posterior in the main network.

In a nutshell, main weights are responsible for ensuring accurate results, while the low-dimensional probabilistic component is responsible for inducing stochasticity to the entirety of the network and stochastically influence the main network weights in a non-linear way. We validate our claims and model across a variety of experimental trials, where we show that our model produces accurate and well-calibrated uncertainty estimates.

1.1.3 Modeling of Weight Correlations in Approximate Inference

Motivation & Problem Statement

Modeling the approximate posterior distributions of deep neural networks is a challenging task. Finding the right balance between efficient optimization, tractable properties, and the ability to capture the true posterior characteristics remains crucial. One inherent characteristic of deep learning networks naturally induces correlations between the neural network weights [11]. This is because modern deep networks leverage techniques like parameter sharing and component reuse to efficiently process high-dimensional data [4]. Recent works have attempted to address this challenge by introducing structured Gaussian approximate distributions. These distributions relax the limitations of diagonal covariance Gaussian distributions, allowing for more flexible covariance structures. However, even simple diagonal covariances can be problematic in some cases. They double the number of learnable parameters while offering, at best, mediocre performance by today's standards.

Contributions

In our work we build upon empirical observations which indicate that under the common setups approximate covariance values across weights converge near to single value. This in combination with the relatively good performance of restricted posteriors [12, 13] lead us to rethink covariance flexibility.

We propose Circulant-Constrained Gaussian as a family of approximate posterior distributions where the flexibility of the covariance matrix (i.e. degrees of freedom) is restricted promoting heavy correlations between network weights. We take advantage of the fact that Circulant matrices can be easily parameterized in terms of their generating kernel, as well as easily manipulated by leveraging their Fourier Transform and related eigendecomposition [14]. Furthermore, our model succeeds in providing a good trade-off between computational complexity and richness of structure. Numerical experiments on a variety of datasets and benchmarks validate our claims. We believe this fact is crucial when deciding on covariance modeling. We argue that practitioners should emphasize capturing the structure of the approximate covariance, even if it means sacrificing some of its flexibility.

1.2 Dissertation Layout

The rest of this dissertation is organized as follows. In Chapter 2, we describe the mathematical foundation for the Bayesian treatment of neural networks. We highlight several methods for applying probabilistic reasoning in these highly non-linear and complex models. Furthermore, we discuss recent techniques and approaches that scale Bayesian inference to deep neural networks. Chapter 3 focuses on the problem of efficient and robust object detection. We present our contribution, which leverages probabilistic feature fusion to achieve this goal. We experimentally validate our approach and provide numerical and visual assessments to substantiate our claims. Subsequently, in Chapter 4, we discuss our contribution to achieving high-performing, low-dimensional Bayesian deep learning methods. We address this challenge by introducing an efficient hypernetwork that injects stochasticity into a primary large neural network. We analyze the pros and cons of our method while experimentally demonstrating its ability to produce highly robust predictions while requiring very low computational cost. Finally, Chapter 5 explores efficient ways to model weight correlation in approximate Bayesian inference. We propose a method that inherently captures low-dependence correlations among individual weights while simultaneously reducing the flexibility of the covariance matrix. Our method is motivated by recent empirical findings and is experimentally validated against more flexible covariance parameterizations. Lastly, Chapter 6 summarizes and concludes this dissertation with a discussion about the future work.

Chapter 2

Bayesian Deep Learning

In this chapter we establish the foundation for the Bayesian deep learning framework. We will start in Section 2.1 introducing the basic elements of deep neural networks and explore their usage in computer vision, highlight some of their drawbacks and limitations. We will continue in Section 2.2 introducing the probabilistic formulation of neural networks and highlight the Bayesian interpretation of standard NN training. Following that, we will build the full Bayesian formulation and discuss its limitations. In Section 2.3 we review some possible ways of approximating the three key-elements of the Bayesian formulation: the posterior (Section 2.3.1), the prior (Section 2.3.3) and the predictive distribution (Section 2.3.2). In Section 2.4 we will review in a more detail the optimization framework for approximating the posterior distribution (Variational Inference). We will discuss different modeling options and their application to deep neural networks. Finally we conclude with Section 2.5 by

exploring methods and frameworks specifically designed and optimized to introduce Bayesian reasoning in deep vision neural networks.

2.1 Neural Networks and Deep Learning

Neural networks have emerged as the cornerstone of modern machine learning, achieving remarkable successes in the past decade [4]. Their widespread use continues to expand in recent years as they have dominated perhaps every field where learning models are applicable. A neural network can be defined as mapping function $g_w : \mathcal{X} \to \mathcal{Y}$ parameterized by parameters $w \in \mathbb{R}^{d_w}$. This function maps input variables $x \subset \mathcal{X}$ to output observed variables $y \subset \mathcal{Y}$ in a non-linear way. Here, w represents the collection of parameters concatenated into a single vector.

One of the simplest mappings can be defined in the form of an affine transformation followed by an element-wise non linear function. A historical example model in terms can be described the logistic regression composed by a linear transformation on the input variables followed by a logistic sigmoid non linear function[15]. Multiple successive composition of such linear transformation layers followed by non-linearities (activation functions) define a neural network (NN) architecture. We now focus on the linear layer as the simplest block in neural networks. For a single layer l it can be defined by its weight matrix W^l and the bias vector b^l:

$$\begin{aligned} h^{(0)} &= x \\ h^{(l+1)} &= \sigma(W^l h^{(l)} + b^l) \\ g_w(x) &= W^l h^{(l)} + b^l \end{aligned} \tag{2.1}$$

We will resort to this notation throughout the dissertation.

Over the years, further transformations have been introduced, expanding the capabilities and applications of neural networks. One of the most successful is the convolutional transformation [16] with the capacity to process high dimensional regularly structured data like images. This transformation lead the way to convolutional neural networks (CNNs). Additionally the notion of recurrence gave rise to recurrent neural networks (RNNs), enhancing network capabilities to handle long-range data dependencies, such as those found in text and audio signals, more properly and efficiently. As for the activation functions, since their introduction, ReLU $\sigma(x) = \max(0, x)$ [17] have largely dominated all other forms of activations primarily due to their ability

to stabilize training dynamics. Albeit ReLU's immersive success other forms of activation functions have been introduced to address different challenges encountered during neural network training. Inspired by neurological signal propagation poling functions [18], (e.g. max-pooling), offer a spatially invariant and robust way to effectively reduce the feature size in the image processing networks. As a way to alleviate optimization difficulties batch normalization [19] was introduced. It normalizes the distribution of a layer's inputs, addressing the issue where parameter changes in one layer can drastically alter the input distribution for subsequent layers. This method's strength lies in integrating normalization from making normalization into the model architecture, enabling the use of much higher learning rates and less sensitive parameter initialization.

In the field of machine learning, *deep learning* refers to the processes of training a neural network with many successive layers. All together the number of their parameters can reach to millions. Modern neural networks of based upon sophisticated modules-layers which we will refer to as building blocks.

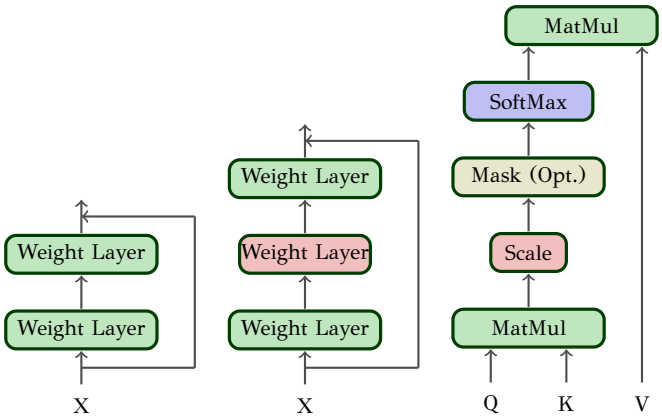

Figure 2.1: Graphical illustration of three common deep learning building blocks: the residual block, the bottleneck residual block, and the scaled dot-product attention block. (From left to right)

These are sophisticated modules carefully designed to tackle specific problems. Examples of modules can be a residual block as set of convolutional transformation with ReLU and skip connection, designed [20] for training stability. The inverted residual blocks [21] designed to reduce feature dimension before applying transformations and non-linearities. The multi-head attention modules which forms the basis of the popular transformer networks [22] and are adept at capturing logn-term relation-

ships between elements in a sequence. As neural networks become more complex, carefully chosen non-linearities become increasingly important. These, along with appropriate initialization procedures, are crucial for training these massive models. Stochastic Gradient Descent, followed by its extension ADAM [23], and RMSprop [24] are optimization algorithms that facilitate the training process in modern deep learning settings, where datasets are often very large.

In the next section, we will describe the probabilistic formulation of deep neural networks by treating the parameters w of $g_w()$ as stochastic. We will highlight the benefits of such formulation and what main drawbacks of standard deterministic training can overcome.

2.2 Probabilistic Perspective of Neural Networks

In this section, we describe the fundamental frameworks of probabilistic learning, the main focus of this dissertation. Specifically, we introduce the main approaches to learning deep neural networks from a Bayesian perspective. We analyze probabilistic modeling (Section 2.2.1), we introduce the maximum likelihood approach for learning parameters (Section 2.2.2), followed by the maximum a-posteriori approach (Section 2.2.3) and finally motivate and introduce the fully Bayesian approach (Section 2.2.4) to learning deep neural networks.

2.2.1 Probabilistic Paradigm

Through out this thesis we consider a supervised learning setting, where we have a training dataset $D = \{X, Y\}$, with inputs $X = \{x_n\}_{n=1}^N$ and outputs $Y = \{y_n\}_{y=1}^N$. In machine learning, we assume that there exist a mapping between variables X and Y that we aim to approximate using a function g with learnable parameters w. Under the probabilistic paradigm, dataset D is defined via a generative processes. Variables x are sampled form the data distribution $p(x)$ and y are formed from by evaluating the conditional $p(y|x)$. The goal becomes to learn the parameters w of a the conditional distribution $p(y|x, w)$. We assume that the mapping parameters w are also probabilistic, so we can say that they follow some ***prior distribution*** $p(w)$ the parameters before the dataset D is observed. Ideally, this should be a distribution over w that induces a distribution over functions $g_w(x)$ which encapsulates all of our

experience and intuition about the problem at hand.

The goal is to learn the parameters w and finally make predictions using the ***predictive distribution*** for new unseen test datum $x^\star$:

$$p(y^\star|x^\star, D) = \int p(y^\star|x^\star, w)p(w|D)dw. \tag{2.2}$$

The evaluation of the predictive distribution assumes the calculation of two major quantities. The first quantity is the ***likelihood function*** $p(y|x, w)$. It represents the probability of observing data point y given an input x and model parameters w. Assuming that each sample from the dataset is independent and identically distributed, the likelihood can be decomposed as individual sample likelihoods:

$$p(D|w) = \prod_{n=1}^{N} p(y_n|x_n, w). \tag{2.3}$$

The likelihood function depends on the nature of the task. In regression problems, the target y is continuous and typically unbounded $y \in \mathbb{R}$ and can be generated as

$$y = g_w(x) + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{2.4}$$

where the random noise is added form a Gaussian distribution with mean zero and variance σ^2. Then the likelihood takes the following Gaussian form:

$$p(y|x, w) = \mathcal{N}(y|g_w(x), \sigma^2) \tag{2.5}$$

where the mean of the Gaussian is estimated via the output of the neural network. Variance parameter σ^2 can also be treated probabilistically with the assignment of proper a distribution. The most common choice is the Gamma distribution which is the conjugate prior for the Gaussian:

$$p(\sigma) = Gamma(\sigma|\alpha, \beta) \tag{2.6}$$

In classification setting where the y variables are K discrete classes.

$$y = \arg\max \ \sigma(g_w(x)) \quad \text{where} \quad \sigma(x) = Softmax(x) \tag{2.7}$$

where the $Softmax()$ inputs model outputs $g_w(x)$ (probits) and transforms them in probabilities over the K classes. The above generative processes results in categorical likelihood functions and the case where we have binary classification problem (i.e. $K = 2$) reduces to the Bernoulli.

$$p(y|x, w) = Categorical(y|Softmax(g_w(x))) \tag{2.8}$$

Again we can define a conjugate function over the softmax probabilities introducing a Dirichlet distribution:

$$p(Softmax(g_w(x))) = Dirichlet(Softmax(g_w(x))|\alpha) \tag{2.9}$$

Where the parameters α can shape the categorical distribution into favoring (assigning more mass) particular classes.

The second term that needs to be evaluated is the ***posterior distribution*** $p(w|D)$. This distribution indicates the probability of w to generate the dataset D. The main focus and difficulty in Bayesian deep learning lies in approximating the posterior. We can further analyze the posterior distribution using Bayes' rule:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} \tag{2.10}$$

The posterior distribution depends on the likelihood, the prior over the parameters. and a third quantity called ***model evidence*** $p(D)$ or marginal likelihood. The second name refers intuitively to the mathematical derivation:

$$p(D) = \int p(D, w)dw = \int p(D|w)p(w)dw. \tag{2.11}$$

The evidence defines the probability we would generate a dataset if we were to randomly sample from the prior over functions $p(g_w(x))$ induced by a prior over parameters $p(w)$.

Evaluating the predictive distribution is inherently coupled with calculation of the posterior. From a probabilistic perspective, the process of learning itself translates to inferring the parameters of the posterior distribution. Direct application of Bayes' rule for this calculation is generally impractical except for some simple cases like linear models and small Gaussian Processes. Several factors hinder acquiring the true posterior. Fist the non-linearities present in neural networks prevent the analytical computation of the involved integrals. Furthermore modern problems involve massive datasets, making efficient evaluation of even closed form solutions prohibitive (e.g. an analytical GP cannot extend to massive datasets). In practical scenarios, the prior distribution over weights and the likelihood function are not conjugate meaning they don't lead to a closed-form solution for the posterior (e.g. multimodal distributions). Finally even if the posterior could be computed for small networks, the same approach wouldn't scale efficiently for larger models. The computational cost grows exponentially with network size, rendering it impractical for modern architectures.

Since we cannot typically compute the true posterior distribution exactly, we resort to approximating it with a tractable distribution, $q(w)$. This approximate distribution should share some characteristics with the true posterior and also lead to well-defined and effective predictive distributions. In the following section, we will review the most common learning settings, maximum likelihood (Section 2.2.2) and maximum a-posteriori (Section 2.2.3) from a Bayesian perspective (i.e., where learning the model is equivalent to learning the posterior) and motivate the use of full Bayesian learning over the previous ones.

2.2.2 Maximum Likelihood

One common way to infer the parameters w is via maximum likelihood (ML) inference. From a Bayesian point of view we can see the ML as a particular choice of prior over parameters and approximate posterior distribution. Specifically the prior distribution is set to a constant value practically being completely uninformative about prior beliefs. Consequently, the true posterior is approximated by the Dirac delta function.

$$\begin{aligned} p(w) &= Const \\ q(w) &= \delta(w - w_{ML}) \end{aligned} \tag{2.12}$$

Where δ is the Dirac function centered around w_{ML} which is a mode of the true posterior distribution $p(w|D)$ under the specific assumptions. More formally:

$$\begin{aligned} w_{ML} &= \arg\max_w \; p(w|D) \\ &= \arg\max_w \; \log p(D|w)p(w) \\ &= \arg\max_w \; \log \prod_{n=1}^{N} p(y_n|x_n, w) + \cancel{\log p(w)} \\ &= \arg\max_w \; \sum_{n=1}^{N} \log p(y_n|x_n, w), \end{aligned} \tag{2.13}$$

where the concave log posterior is being optimized as it is easier and numerically stable. The log prior is canceled as constant and doesn't affect the optimization. Finally the predictive distribution of Eq. 2.2 takes the following form:

$$\begin{aligned} p(y^\star|D, x^\star) &= \int p(y|x^\star, w)q(w)dw \\ &= p(y|g_{w_{ML}}(x^\star)). \end{aligned} \tag{2.14}$$

Unfortunately, neural networks trained via maximum likelihood can be prone to overfitting. In this scenario the network obtains increasingly good performance on the training set, but begins to deteriorate in its predictive performance on unseen test data. This behavior is coupled with the over-parameterization. The modern neural networks have way more parameters being able to memorize entire datasets as [25] demonstrated that deep neural networks have sufficient capacity to fit randomized labels on popular image classification tasks.

Many methods have been proposed to prevent overfitting, including early-stopping. This technique halts the optimization of the weights before the loss function reaches a minimum. By doing so, the model doesn't have time to overfit the training dataset, leading to better performance on unseen data. Other methods focus on limiting the complexity of the network, such as reducing its depth or the number of hidden units. This approach aligns with Occam's razor principle, which favors simpler models whenever possible [26]. Large neural networks, offer a larger support (ability to represent complex datasets). However we still need to carefully introduce priors on the parameters to ensure the model learns functions with reasonable inductive biases. [27]

2.2.3 Maximum A-Posteriori

Maximum A-Posteriori (MAP) inference is as a natural extension of the ML framework. In MAP we incorporate prior beliefs about the desirable properties of the NN functions, which are encapsulated in $p(w)$ the prior distribution over weights. The approximate posterior remains a delta function centered at the posterior mode.

$$\begin{aligned} p(w) &= \mathcal{N}(0, \hat{\sigma}^2) \\ q(w) &= \delta(w - w_{MAP}). \end{aligned} \tag{2.15}$$

Here we specified $p(w)$ is set to a zero mean Gaussian distribution with a hyper-parameter $\hat{\sigma}^2$, a modeling choice which heavily linked to $L2$ regularization and the

weight decay method.

$$
\begin{aligned}
w_{MAP} &= \arg\max_{w} \; p(w|D) \\
&= \arg\max_{w} \; \log p(D|w)p(w) \\
&= \arg\max_{w} \; \log \prod_{n=1}^{N} p(y_n|x_n, w) + \log p(w) \\
&= \arg\max_{w} \; \sum_{n=1}^{N} \log p(y_n|x_n, w) + \log p(w).
\end{aligned}
\tag{2.16}
$$

Consequently, there are several choices for prior distribution specification (see Section 2.3.3). Some popular choices include sparse priors, which are members of the family of multivariate scale mixtures of Gaussians. These distributions are closely related to widely used approaches for sparse deep learning, including, among others, he $L2$ regularization, Laplacian priors ($L1$ regularization) [28] also known as LASSO in the context of linear regression and Student-t priors (e.g. the relevance vector machines) [29]. Introducing sparse priors into the learning process encourages networks with smaller parameter magnitudes. This is desirable because networks with large weights tend to represent more complex functions, often exhibiting greater oscillation as a function of inputs [30]. Notably, the majority of deep neural networks are trained using the maximum a-posteriori (MAP) estimator.

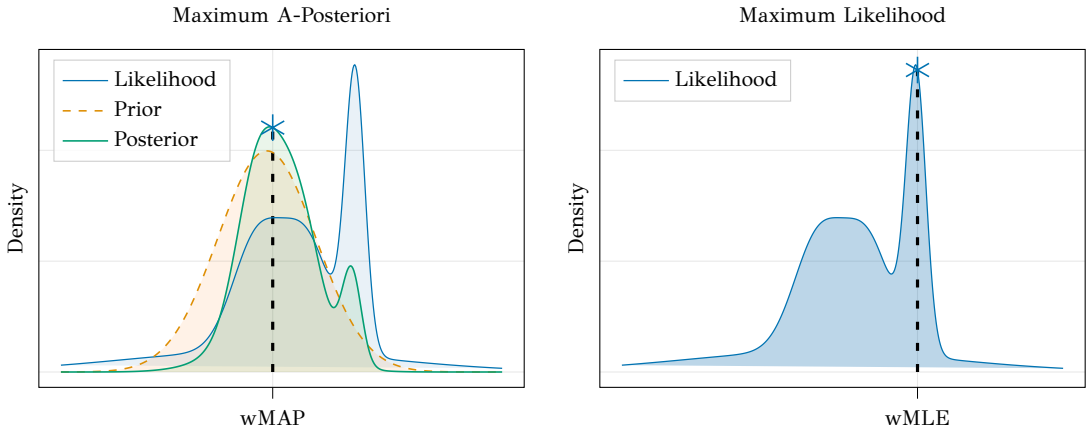

Figure 2.2: Illustration of how the introduction of a non constant prior distribution reshapes the optimization landscape resulting in a different solution w for MAP (left) inference compared to ML (right).

Figure 2.2 provides an illustrative example of how MAP could lead to better model solution than ML inference. Specifically we can observe the densities of the

likelihood and the posterior and how the introduction of a prior shapes the optimization landscape. In the left plot, the likelihood function (equivalent to the posterior in the ML case) with a local maximum at w_{MLE}. The right plot shows how introducing a prior function reshapes the MAP posterior, leading to a new objective function that drives the maximization towards a different set of values. In the MAP case, the right choice of prior helps the optimization to achieve a flatter maximum compared to the potentially overfitting one found by ML. Sharp minima are prone to non-robustness against training-test distribution shifts, leading to poor performance on unseen data [31].

Even with the introduction of suitable prior distribution it has been noticed that modern neural networks are often miscalibrated. This means that their predictions are typically overconfident [32]. For example, in classification the highest softmax output of a convolutional neural network is typically much larger than the probability of the associated class label. The fundamental reason for miscalibration is restricting the posterior to Dirac having only deterministic point estimate solution of parameters.

2.2.4 Bayesian Inference

In Bayesian inference, we move beyond the deterministic approximation of weights used in MAP and ML estimation. Instead, we aim to obtain a distribution for the weights themselves. Therefore, the learning process becomes one of estimating a distribution, $q(w) \approx p(w|D)$, that closely approximates the true posterior distribution. Neural networks with probabilistic weights are called *Bayesian Neural Networks* (BNNs). The key advantage of these models compared to standard point-estimated counterparts is the fact that they produce *uncertainty* estimates associated with the networks' outputs.

The main uncertainty encountered in the NN predictions of BNNs can be decomposed to the following two sources:

$$p(y|x, D) = \int \underbrace{p(y|x, w)}_{\substack{\text{Data} \\ \text{Uncertainty}}} \underbrace{q(w|D)}_{\substack{\text{Model} \\ \text{Uncertainty}}} dw \tag{2.17}$$

The first source of uncertainty is the data uncertainty or *aleatoric* uncertainty. This is arises from noise inherent in the true underlying generative process of the data. For example, noise on an image captured by a faulty measurement system [33]. An illustrative example is the regression modeling in Eq 2.4, where aleatoric uncertainty is

modeled with independent normal noise (ϵ). Notably, even large amounts of data may not be sufficient to fully capture this type of uncertainty. On the other hand model uncertainty also known as *epistemic* uncertainty, captures our uncertainty about the specific set of weights that best explains the data (multiple sets can lead to excellent results). This uncertainty about the parameters is then propagated into the predictions. Bayesian neural networks provide a natural and principled way to account for the epistemic uncertainty as they would automatically consider and ensemble all the possible settings of the parameters that are consistent with the training data.

In other words, by marginalizing over the parameters during prediction, we propagate our epistemic uncertainty about the model weights to the predicted outputs. This epistemic uncertainty allows these models to potentially provide better predictions and uncertainty estimates compared to standard training procedures. This additional information can then be leveraged to improve the reliability of neural networks in tasks such as predictive uncertainty calibration, out-of-distribution data detection, and safety-critical applications. The ability to marginalize over different weights addresses issues of miscalibration and overconfidence of standard models [34].

To optimize a distribution, we need to establish a framework for acquiring values that approximate the true posterior distribution based on our prior beliefs. In the next section, we will establish the theoretical foundation for Bayesian posterior approximation and focus on popular methods for this task.

2.3 Approximate Inference

Building on the benefits of Bayesian treatment for neural networks, in this section we explore methods to address the challenges encountered in application and training of BNNs. While this might seem straightforward in theory, direct application of Bayesian inference becomes very challenging in practice. Here, we examine established practices and techniques for approximate Bayesian inference. Specifically, this section focuses on the three main probabilities encountered in Bayesian inference, as discussed in Section 2.2.1. First, Section 2.3.1 introduces methods for approximating the posterior distribution. Subsequently, Section 2.3.2 delves into the challenges associated with the marginalization process for the predictive distribution. Finally, Section 2.3.3 concludes

with a discussion on the prior distribution and highlights strategies for selecting appropriate distributions for neural network weights.

2.3.1 Posterior Approximation

In this subsection, we discuss and summarize some of the most common and popular methods for approximating the posterior distribution. In the Bayesian framework of learning, the ultimate goal is to find an accurate and meaningful approximation of the posterior distribution. This has motivated extensive research to develop and propose new methods. We begin by discussing the gold standard in probabilistic learning: Markov chain Monte Carlo (MCMC) methods. Unlike other methods, MCMC does not approximate the posterior; instead, it attempts to directly sample from the true distribution. Next, we will explore methods that perform local approximations of the distribution around a specific mode (maximum) of the true posterior. These methods are known as Laplace approximation methods. Finally, we will discuss entirely new approaches specifically designed to address the challenges of deep learning and the high dimensionality of modern networks.

Markov Chain Monte Carlo & Sampling Methods

Rather then trying to find tractable forms of the integral in Eq 2.2, one can often use numerical integration techniques based on discretisation and Monte Carlo. A Monte Carlo approach would make use of the property that the integral is computed against a probability distribution. This allows the expectations to be approximated by a finite sum.

$$\int f(w)p(w|D)dw \approx \frac{1}{K}\sum_{k=1}^{K} f(w_k) \tag{2.18}$$

As long as the samples are drawn from the distribution $p(w|D)$ the estimation has the same mean. Unfortunately, this simple Monte Carlo approach assumes that it is easy to draw samples from the posterior, which is not true in most cases.

Sampling-based approaches are historically one of the most fundamental methods for probabilistic inference. These methods, unlike those that attempt to directly approximate the posterior distribution, aim to acquire samples from the *true posterior distribution*. The main framework generates probable samples from a target distribution relying on a proposal distribution. This is a surrogate distribution from which

samples can be easily drawn. These base samples can be transformed using importance sampling techniques, or rejected according to our knowledge about the target distribution (rejection sampling). Unfortunately, these methods (as discussed in [15]) suffer from severe limitations, particularly in high-dimensional spaces.

To address limitations in dimensionality and computational efficiency *Markov chain* approaches have been introduced. These methods are based in a proposal distribution $g()$. This time, however, they build a framework for maintaining a record of drawn samples which affect the proposal distribution () which depends the sequence of samples $w^{(1)}, w^{(2)} \dots w^{(n)}$ which form a Markov chain. The Metropolis-Hastings algorithm provides a solid framework upon the majority of modern sampling methods and approaches are based. In the case of the Bayesian setting the true posterior distribution can be approximated as $q(w|D) \propto p(D|w)p(w)$ simply ignoring the normalizing constant. The algorithm simulates samples from the true $p(w|D)$ using a proposal distribution $g(w_1|w_2, D)$, βυ fist sampling $w^\star$ from the proposal distribution and then accepting the sample with a with the probability $\min(1, r)$ where:

$$r = \frac{q(w^\star|D)g(w^{(t)}|w^\star, D)}{q(w^{(t)}|D)g(w^\star|w^{(t)}, D)} \tag{2.19}$$

Extending the above framework Hamiltonian (Hybrid) Monte Carlo (HMC) [35, 36] provides a more sophisticated method for proposing samples of w in a Metropolis-Hastings framework. Compared to standard random-walk proposals, HMC efficiently explores the state space.

Despite their effectiveness, Hamiltonian Monte Carlo (HMC) methods can be computationally expensive when applied to modern deep neural networks. To address this challenge, recent work has proposed Scalable Gradient HMC (SGHMC) as a more scalable variant [37]. However, high-fidelity sampling methods remain valuable only in research. These methods, as exemplified by [38], provide a reliable way to assess assumptions about approximations and establish baselines for more efficient approaches. Notably, works like [38] have even attempted using HMC on large-scale CNNs to explore fundamental questions in Bayesian deep learning.

Laplace Approximation

One of the most common methods for approximating the posterior distribution is Laplace approximation. While originally proposed by Mackay in [30], it has recently seen a resurgence in popularity due to its ability to work with modern large-scale

neural networks and datasets (e.g., [39]). The core idea is to approximate the true posterior distribution of the parameters locally, around an already acquired mode (i.e., the most likely value) of the posterior. This "post-hoc" approach is particularly appealing in deep learning because it can leverage the vast number of pre-trained models available to researchers.

Specifically, the Laplace approximation utilizes the second-order Taylor series expansion of the loss function $\mathcal{L}$ around the mode w_{MAP} approximating in this way the posterior distribution $p(w|D)$. More specifically the loss can be approximated as follows:

$$\mathcal{L} \approx \mathcal{L}(D, w_{\text{MAP}}) + \frac{1}{2}(w - w_{\text{MAP}})^T(\nabla_w^2 \mathcal{L}(D, w)|_{w_{\text{MAP}}})(w - w_{\text{MAP}}). \tag{2.20}$$

The term in the expansion accounting for the first-order derivatives is canceled out because the expansion point, by definition, is the mode of the distribution. $\nabla_w \mathcal{L}(D, w)|_{w_{\text{MAP}}} = 0$. In fact the mode for the true posterior can be obtained by via MAP estimation 2.2.3. Taking the exponential of Eq. 2.20 we obtain an normalized distribution $q(w)$ that results to Gaussian distribution with mean the mode w_{MAP} and and precision proportional to the negative inverse Hessian of the model parameters:

$$p(w|D) \approx q(w) = \mathcal{N}(w_{\text{MAP}}, \Sigma) \quad \text{where} \quad \Sigma = \left(\nabla_w^2 \mathcal{L}(D, w)|_{w_{\text{MAP}}}\right)^{-1}. \tag{2.21}$$

In general, any prior distribution with twice differentiable log-density can be used. But due to its popularity a zero-mean isotropic Gaussian prior with precision γ is assumed in the majority of cases. This leads to the following:

$$\nabla_w^2 \mathcal{L}(D, w)|_{w_{\text{MAP}}} = -\sum_{n=1}^{N} \nabla_w^2 \log p(y_n|g_w(x_n))|_{w_{\text{MAP}}} - \gamma^{-2} I. \tag{2.22}$$

One of the main drawbacks of the plain Laplace approximation is the computational cost associated with calculating the Hessian. As in any application where information about the second derivative of the parameters is required, practitioners often resort to approximations for the Hessian structure. In the remainder of this section will briefly introduce advances in Hessian approximations and their factorizations.

At this point, we observe that there is no guarantee that the direct computation of the Hessian in Eq. 2.22 will result to a positive definite matrix. The Hessian is then approximated with the Generalized Gauss-Newton (GNN) matrix [40]:

$$G = \sum_{n=1}^{N} J(x_n)(\nabla_g^2 \log p(y_n|g)|_{g=g_{w_{\text{MAP}}}}(x_n))J(x_n)^T. \tag{2.23}$$

Where $J(x_n)$ is the Jacobian matrix derived as:

$$J(x_n) = \nabla_w g_w(x_n)|_{g=g_{w_{\text{MAP}}}}. \tag{2.24}$$

The Gauss-Newton matrix is also interconnected with the Fisher information matrix. However, since both the GNN and the Fisher information matrix are still quadratically large (i.e., as large as the exact Hessian of the network), their computation is often infeasible [40].

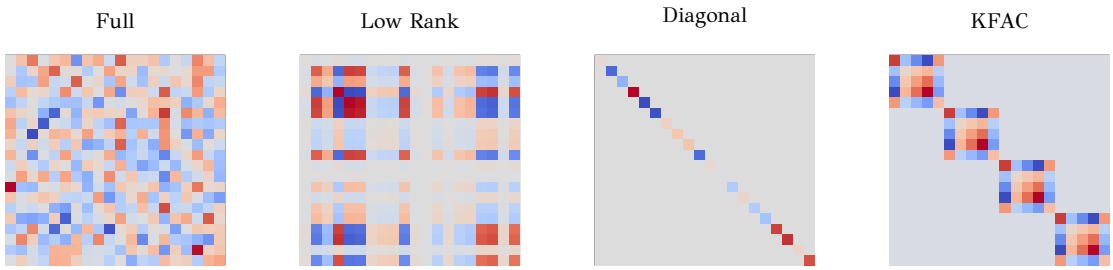

Figure 2.3: Common Generalized Gauss-Newton matrix lightweight parameterizations.

To circumvent the quadratic computational cost of evaluating the GGN matrix several parameterizations have been proposed. The most straightforward approach is the diagonal restriction of the matrix, where all the off-diagonal elements are set to zero. Recent work has shown that this factorization can be effective for sufficiently deep neural networks [41]. This finding suggests that even simple posterior approximations can sometimes be sufficient as neural network depth increases significantly. Beyond reducing the storage cost of the matrix, the diagonal covariance structure also makes its inversion highly computationally efficient.

More expressive alternatives include block-diagonal factorization methods such as Kronecker-factored approximate curvature (KFAC) [42]. This parameterization has been tested extensively in the large-scale NNs [43]. The key idea is to model the correlation between weights in the same layer, while assuming independence between weights from different layers. This is a more realistic assumption from than the diagonal one as recent works suggest [11]. The Kronecker factorized parameterization leads to block diagonal structure of the GGN which can be intrinsically viewed as a Matrix Normal distribution over the weights [44]. To improve KFAC's efficiency, its low-rank factorization can be considered [45]. This approach involves eigendecomposing the Kronecker factors and retaining only the eigenvectors corresponding to the

first k largest eigenvalues. We can go beyond approximating individual blocks of the Hessian or GGN with low-rank structures, as the entire Hessian or GGN itself can also be approximated using a low-rank structure [46]. These advancements have made Laplace approximation-based methods truly applicable to a wide range of models and problems [47, 48, 9].

Stochastic Weight Averaging

In this subsection, we discuss approximate Bayesian inference methods that heavily rely on extracting valuable information from the stochastic gradient decent (SGD) training trajectory of neural networks. One of the pioneer of using SGD characteristic where [49] proposed to use the iterates of averaged SGD as an MCMC sampler. After analyzing the dynamics of SGD using tools from stochastic calculus show that SGD has a Gaussian limiting distribution. Building on this notion, authors in [50] found that simple averaging of multiple points along the trajectory of SGD, with either a cyclical or constant learning rate, leads to better generalization performance than standard training.

This trajectory-based learning approach is called stochastic weight averaging (SWA). Following this method, weights of the network obtained after epoch i as w_i, the SWA solution after T epochs is given by:

$$w_{\text{SWA}} = \frac{1}{T}\sum_{i=1}^{T} w_i. \tag{2.25}$$

Unlike SGD, which converges to a single point in the weight space, SWA explores a set of possible solutions. This approach leads to finding wider local optima, which translates to better generalization [50]. Building upon this work, Maddox et al. [10] extended the procedure by attempting to approximate the empirical second moment of the stochastic SGD trajectory. This approximation is used as an estimate for the true posterior distribution's second moment at test time. The simplest approximation to the second moment of the distribution of weights is to restrict the empirical trajectory covariance matrix to be diagonal

$$\Sigma_{\text{diag}} = \text{diag}(\bar{w^2} - w_{\text{SWA}}^2) \quad \text{where} \quad \bar{w^2} = \frac{1}{T}\sum_{i=1}^{T} w_i^2. \tag{2.26}$$

Where the running average of the second moment is maintained for each weight along side the their mean value. Further more one can easily acquire a structured

variation, enriching the rather restricted diagonal covariance approximate matrix. A low-rank component can be integrated to the diagonal approximation allowing for cross weight correlations in the final model

$$\Sigma_{\text{low rank}} = \frac{1}{T-1}\sum_{i=1}^{T}(w_i - \bar{w_i})(w_i - \bar{w_i})^T. \tag{2.27}$$

Where the $\bar{w_i}$ is an approximate of the w_{SWA} at the current epoch i. The resulting matrix is of rank T and can be combined with the diagonal one to yield the final posterior approximation distribution.

$$p(w|D) \approx q(w) = \mathcal{N}(w_{\text{SWA}}, \Sigma_{\text{diag}} + \Sigma_{\text{low rank}}) \tag{2.28}$$

This trajectory averaging approach has produced quite excellent experimental results. This property, along with its simplicity and ease of implementation, has made it a standard baseline method for approximating the posterior distribution in Bayesian deep learning research within a short period of time.

Further Approximations

In this subsection, we will discuss alternative methods for approximating the posterior distribution that deviate from those previously discussed. These methods include ensemble techniques, which have roots in bootstrapping models, but are not strictly Bayesian according to the rigorous mathematical definition. Furthermore we will also explore the use of neural networks to directly acquire samples from the posterior distribution.

Posterior Networks: are a recently developed class of approximate Bayesian inference methods. These networks utilize neural networks to directly acquire samples of epistemic uncertainty, deviating from traditional probabilistic modeling approaches. While this approach may seem to diverge from the established direction and narrative of this section, the potential for a more straightforward approach to probabilistic reasoning in modern deep neural networks compels us to mention them briefly. We will describe these methods in the context of classification. Here, the categorically distributed class label $y_i \sim \text{Cat}(p_i)$ is modeled with its natural conjugate prior, the Dirichlet distribution $q(i) = \text{Dir}(\alpha_i)$. The key idea is to use a neural network to directly predict the parameters of the Dirichlet distribution, rather than predicting the parameters of the categorical distribution (i.e., converting logits to class probabilities)

[51]. However, challenges arise during the training of these networks, particularly in acquiring useful epistemic uncertainty estimates. The Dirichlet prior needs to be shaped differently for in-distribution data, where its mass can be assigned to the true class label. Conversely, it must retain a more spread mass for out-of-distribution data (input data that deviate significantly from the training data). Many works directly model this property by incorporating out-of-distribution (OOD) data during training to smooth the Dirichlet probability space. However, this requirement for OOD data is often impractical. A recent class of networks proposed in [52] bypasses the need for OOD data. It learns the prior distribution parameters by first mapping input samples x to a latent space z. Then, a parametric distribution $p(z|c)$ is trained for the class c. demonstrate that by learning a latent mapping, a manifold naturally forms clusters. This allows $p(z|c)$ to assign the appropriate mass to each class for each sample z_i.

Deep Ensembles: or ensembling neural network predictions is a well-known practice that leads to improved generalizability of predictions [53]. Predictions from different ensemble members are combined to create a form of predictive distribution in the notion of Eq. 2.2 [54]. Recent work has shown that deep ensembles can outperform some Bayesian approaches for uncertainty estimation, making them a strong baseline for comparison [55]. Although ensemble methods do not strictly adhere to the mathematical formulation of Bayesian inference, they can be considered a compelling approach to approximating Bayesian model averaging [56]. This is because ensembles, by their nature, can account for multiple parameter solutions, which aligns with the concept of a multimodal true posterior distribution. In simpler terms, rather than relying solely on a single hypothesis with a specific set of parameters w, these methods leverage all possible parameter settings, weighted by their corresponding posterior probabilities [27].

Further research has explored enforcing neural network diversity, in an afford to capture multiple ”posterior modes”. This is achieved by making individual models within the ensemble repel each other in weight space [57]. This ensures no two models share identical weights. Directly enforcing diversity [58] by encouraging the networks to produce discrepancies between its predictions could lead to sub-optimal performance [59]. Since diversity on highly accurate models can only come at the expense of predictive performance. The greatest drawback of deep ensemble methods is their computational cost as the improved predictive performance and calibration are strongly tied to the number of networks in the ensembles.

2.3.2 Predictive Approximation

Having discussed various options for evaluating the approximate posterior distribution, this section explores some fundamental approaches to approximating the predictive distribution. Even with the posterior distribution available, marginalization in Eq. 2.2 is not a straightforward procedure. As such, we still need to resort to approximations. We will begin in Section 2.3.2 by introducing the Monte Carlo estimator as the most straightforward method for approximating the predictive distribution. Subsequently, Section 2.3.2 details the linearization approaches. We will then continue with variance and uncertainty propagation techniques in Section 2.3.2, concluding in Sec. 2.3.2 with a discussion about distillation approaches.

Monte Carlo

Definitely the most popular way of approximating the predictive distribution is the Monte Carlo (MC) iteration algorithm:

$$p(y^\star|x^\star, D) \approx \frac{1}{K}\sum_{i=1}^{K} p(y|x^\star, w_i) \quad \text{where} \quad w_i \sim q(w|D) \tag{2.29}$$

This straightforward approach, while easy to implement, yields unbiased estimates. The mean of the summation approximates the true integral with an error of $1/\sqrt{K}$, but only as K approaches infinity. In practice, dealing with large K values is infeasible. Therefore, we resort to using smaller K values for efficient inference on large models. However, these small sample sizes compared to the high dimensionality of neural networks can introduce bias in the final integration outcome. Despite this, MC-based predictive distributions can still achieve state-of-the-art results [55].

Linearization

Monte Carlo integration requires evaluating the neural network multiple times for every new test point prediction. This can be computationally expensive and prohibitive for real-time applications. Linearization offers an alternative. It provides an analytical approximation for Eq. 2.2, eliminating the need for exhaustive re-evaluations in MC iterations. This processes starts by applying the Taylor approximation to the NN

output $g = g_w(x) \in \mathbb{R}^s$ around the point weights μ:

$$\begin{aligned} g_w(x) &\approx g_\mu(x) + \mathcal{J}(x)(w - \mu) \\ \text{where} \quad \mathcal{J}(x) &= \nabla_w g_w(x)|_\mu. \end{aligned} \tag{2.30}$$

The matrix $\mathcal{J}(x) \in \mathbb{R}^{s \times d}$ is the Jacobian matrix, holding the partial derivatives of the $g_\mu(x)$ the neural network evaluated at the μ w.r.t. w. This approximation allows the neural network output to linearly depend on the w weights. To evaluate the resulting distribution on the NN outputs $p(g)$, we first note that g depends linearly on w. Under the Gaussian assumption $q(w) = \mathcal{N}(\mu, \Sigma)$ the NN weights w can be marginalized:

$$p(g|x) \approx \int \delta(g - g_w(x))q(w)dw = \mathcal{N}(g|g_\mu(x), \mathcal{J}(x)^T \Sigma \mathcal{J}(x)). \tag{2.31}$$

The computation of this distribution is not only useful for theoretical analysis, but it also simplifies the inference problem. As inference involves reasoning about the weight vector w of dimensionality d. This approach is replaced by evaluating $g(x)$, which has a dimensionality of $s << d$. This shift is advantageous because higher dimensionality often leads to computational difficulties.

Finally the computation of the predictive distribution Eq. 2.2 can be defined as:

$$p(y^\star|x^\star) = \int p(y|g^\star)p(g^\star|x)dg. \tag{2.32}$$

The Gaussian weight assumption makes this computation tractable for Gaussian likelihood functions $p(y|g)$ (i.e. regression). However, for non-Gaussian likelihoods, especially in classification, the introduction of the Softmax or sigmoid function renders direct integration intractable. Fortunately, the Gaussianity of the neural network outputs allows us to obtain a good approximation ([60] [26] [61] by exploiting the close similarity between the logistic sigmoid function $\sigma(a)$ and the probit function $\Phi(a)$ leveraging the analytical solutions of the probit approximations.

Linearized predictive models have gained significant popularity due to their ability to improve predictive performance, particularly for common underfitting problems encountered with the Laplace approximation [62]. However, the computational cost associated with Jacobian computation restricts their application to small networks, output spaces, and datasets. Works have addressed this limitation by introducing a scalable, sample-based Bayesian inference method for conjugate Gaussian multi-output linear models [48]. Additionally, potential pitfalls have been discussed in the context of Bayesian optimization and active learning [39].

Variance Propagation

Variance propagation is an alternative method that can be used to obtain a distribution for the network's output. This approach avoids the use of Monte Carlo sampling, thus reducing variance in the likelihood. Variance propagation methods perform approximate inference analytically by propagating means and variances of the weights through all layers of a neural network to the output while ensuring computational efficiency and differentiability. Several authors have proposed methods that propagate moments in different contexts. Probabilistic Back Propagation [63] utilizes moment matching principles from the expectation propagation framework [64] to compute the approximate posterior.

To illustrate moment propagation, we will focus on recent sampling-free approaches based on the central limit theorem (CLT). These approaches typically omit covariances in the second moment. The core principle of moment propagation based on the central limit theorem property is to focus on a single linear layer, denoted as l, with weights $W^l \in \mathbb{R}^{I_l \times O_l}$, pre-activations $\alpha^l \in \mathbb{R}^{1 \times I_l}$, and activations $h^l \in \mathbb{R}^{1 \times \iota}$.

$$\begin{aligned} h^l &= f(\alpha^{(l-1)}) \\ \alpha^l &= W^l h^l + b \end{aligned} \tag{2.33}$$

Since the linear transformation is applied to h^l after the non-linear function this results the pre-activations α using the CLT being approximately normally distributed. Mean and variance parameters can be computed by moment matching from the earlier layers as:

$$\begin{aligned} \mathbb{E}[\alpha^l] &= \mathbb{E}[W^l]\,\mathbb{E}[h^l] + \mathbb{E}[b] \\ \text{Var}[\alpha^l] &= \text{Var}[h^l]\text{Var}[W^l] + \text{Var}[h^l]\,\mathbb{E}[W^l]^2 + \mathbb{E}[h^l]^2\text{Var}[h^l]. \end{aligned} \tag{2.34}$$

The mean and variance of the post-activation feature maps $h = \text{ReLU}(\alpha^{(l-1)})$ are tractable for ReLU activations [65]. Here, is often assumed that the pre-activations α^l can be approximated by a Gaussian distribution with mean μ and variance σ^2. Under this assumption, the mean and variance of the output distribution through the non-linearity become tractable.

$$\begin{aligned} \mathbb{E}[\text{ReLU}(\alpha^{(l-1)})] &= \mu\Phi\left(\frac{\mu}{\sigma}\right) + \sigma\Phi\left(\frac{\mu}{\sigma}\right) \\ \text{Var}[\text{ReLU}(\alpha^{(l-1)})] &= (\mu^2 + \sigma^2)\Phi\left(\frac{\mu}{\sigma}\right) + \mu\sigma\phi\left(\frac{\mu}{\sigma}\right) - \mathbb{E}[h]^2. \end{aligned} \tag{2.35}$$

Propagating the covariances has been studied more thoroughly in [66], where the authors focus on the effect of activation covariance propagation. Furthermore, in

recent years, numerically stable approximations have been developed for propagating means and variances through multivariate functions such as softmax, argmax, and log-sum-exp. These approximations can handle categorical distributions as well as non-linearities like max-pooling and leaky ReLU [67]. Additionally, approximations have been developed to deal with batch normalization layers in neural networks, as presented in [68].

Expectation Distillation

Another popular way to approximate the predictive distribution is through the process of distillation. Following this idea the predictive distribution under different sampled parameters $w_i \sim q(w)$ can be learned or "distilled", into single neural network with ϕ parameters. This method offers the benefit of requiring only one evaluation at test time, in contrast to Monte Carlo integration. Among the first to apply this were the [69] who proposed the Bayesian-dark-knowledge framework. This framework is a way to train a parametric model $\mathcal{S}(y|x,\phi)$ to approximate the Monte Carlo posterior predictive distribution $q(y|x)$ in order to gain the benefits of the Bayesian approach while only using the same run time cost as the plugin method. Following [70], $q(y|x)$ is called the "teacher" and $S_\phi(y|x)$ the "student". The framework of knowledge distillation uses stochastic MCMC methods (see Section 2.3.1) to estimate $q(w)$ and hence $q(y|x)$ online. Then uses these estimates to train the student network to minimize the KL divergence between the teacher and the student predictions:

$$\begin{aligned}\mathcal{L}(\phi) &= \mathrm{KL}(p(y|x,D)||\mathcal{S}_\phi(y|x)) \\ &= -\mathbb{E}_{p(y|x,D)}\left[\log \mathcal{S}_\phi(y|x)\right] + \text{const} \\ &= -\int \left[\int p(y|x,w)p(w|D)dw\right] \log \mathcal{S}_\phi(y|x)dy \\ &= -\int p(w|D)\left[\mathbb{E}_{p(y|x,w)} \log \mathcal{S}(y|x,\phi)\right] dw.\end{aligned} \tag{2.36}$$

Unfortunately, computing this integral is not analytically tractable. However, we can approximate this by Monte Carlo

$$\mathcal{L}(\phi) = -\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{p(y|x,w)} \log \mathcal{S}(y|x,w) \tag{2.37}$$

$$w_i \sim p(w|D) \approx q(w) \tag{2.38}$$

This technique of training a neural network to approximate an MC expectation and deploying it at test time has been further developed. Works like [71] make

connections and utilize this approach to distill Deep Ensembles [72] as an expectation of networks trained under different random seeds. A shortcoming of the distillation approach is that it only distills the posterior predictive distribution, thus losing access to other valuable posterior statistics. However, a more significant limitation regarding uncertainty estimation is the loss of information about the ensemble's diversity. This results in making the posterior distillation performance highly sensitive to the student model's architecture.

2.3.3 Prior Specification

The prior distribution summarizes our beliefs about the neural network parameters before we even see any data. Specifying a particular distribution can be very challenging, as it's not straightforward to set specific values for the neural network parameters. This makes defining a Bayesian prior notoriously difficult for complex models like neural networks. The high dimensionality of parameter space further complicates reasoning about the NN parameters. Since the true posterior distribution can almost never be recovered, it is difficult to isolate the influence of a prior (even empirically) [27]. Additionally simple priors might be sufficient, as neural networks possess strong inductive biases due to their architectural structure. In other words, simple priors have the potential to induce highly complex functions, which is often more important in applications than using more complex and informative parameter priors. This section discusses the two main approaches to prior specification in Bayesian inference. We first analyze common priors placed directly on neural network parameters (Section 2.3.3). Subsequently, in Section 2.3.3, we will briefly discuss priors that implicitly affect the network weights by acting upon each layer's activations.

Parameter Space Priors

Since reasoning about specific network weight values is often impractical, the majority of Bayesian inference methods employ a standard Gaussian distribution as the prior. Therefore, the isotropic zero-mean Gaussian prior, $p(w) = \mathcal{N}(0, I)$ is the most common choice. Most practitioners chose this prior for its simplicity, heavily relying on better and more flexible posterior approximation to capture the true posterior distribution and achieve improved predictive performance and uncertainty quantification in their

models.

Instead of promoting specific weight values, parameter space priors have been employed to enforce desirable attributes on the learned weights. One of the most fundamental properties is sparsity. Sparsity not only leads to lighter models by pushing neural network values closer to zero, but it has also been shown to improve model robustness and generalization capabilities [73, 74]. A common approach to enforcing sparsity is through sparse priors. These distributions can be implemented using hierarchical modeling with a non-centered parameterization.

$$w \sim \mathcal{N}(0, z^2) \quad \text{where} \quad z \sim p(z)$$
$$p(w) = \int \mathcal{N}(0, z^2) p(z) dz$$

By treating the scales z of w as random variables we can recover marginal prior distributions over the parameters that have heavier tails and more mass at zero; this subsequently biases the posterior distribution over w to be sparse. An example of resulting w priors for different choices of scale distributions is illustrated in Fig 2.4. The hierarchical modeling framework results in heavy-tailed priors. These priors

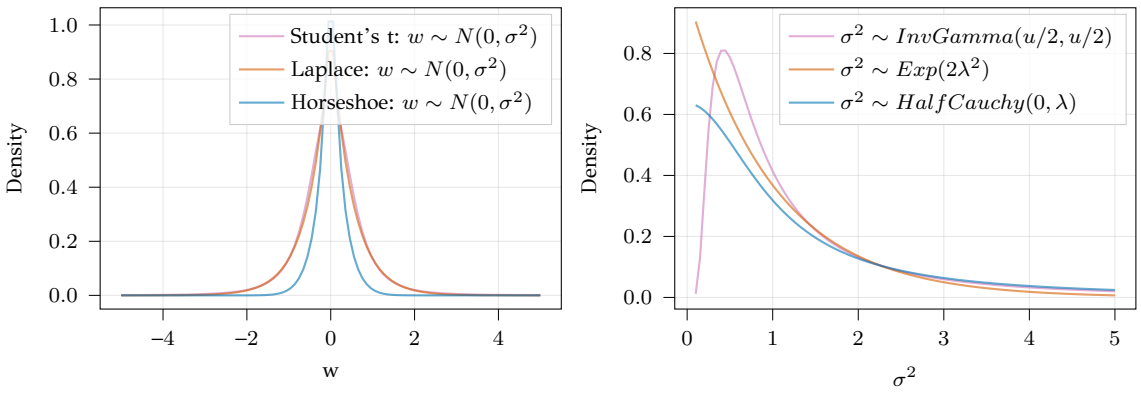

Figure 2.4: Hierarchical modeling of common sparse priors.

apart from shifting the majority weights to zero, give the model freedom to converge in large weight values as well; allowing for more model flexibility.

Furthermore, some works propose determining priors over parameters by exploiting empirical weight properties of already ML trained networks. For example, Fortuin et al. [11] proposed structured Gaussian priors based on empirical weight values, capturing the empirical correlations in neural network weights. Priors for invariance can be another way to specify informative prior distributions. Authors in [75] proposed learning a model that exploits invariance in model predictions under

certain input transformations. Finally, [76] explored how Gabor filters can be specified as an implicit probability distribution for the first layer filters of a convolutional neural network (CNN). This approach leverages the well-known properties and empirical distribution of already learned neural network parameters.

Function Space Priors

Rather than working with the uninterpretable parameter space, function space priors place the Bayesian prior on the model's induced functions directly. As we truly care about the distribution over predictive functions, specifying functional priors has received much attention [77]. Placing a functional prior on a neural network requires either taking infinite-width limits [78] or optimizing divergences involving stochastic processes, which are distributions over functions. An intuitive and well-studied case is Gaussian processes (GPs), which define a distribution over functions characterized by a mean and a kernel function. Sun et al. [79] proposed a framework extending variational inference (see Section 2.4) to minimize the divergence between two distributions over functions via a function-space evidence. This allows the introduction of GP priors in BNNs and the integration of more interpretable priors.

To avoid the infinite support of stochastic processes and the inaccuracies associated with directly inferring and working in the function space, especially in high dimensions, several methods have been proposed. These methods implicitly define a prior in the function space while avoiding the computational burden of directly working in that space. An illustrative example is the function-space regularization method presented by [80]. The authors take a probabilistic perspective on function-space regularization in deep neural networks, defining an empirical prior distribution (in the Maximum A Posteriori style) over parameters. This allows for the explicit encoding of relevant prior information about the data-generating process into the training process. Furthermore, Tran et al. [81] proposed finding weight parameter values that produce GP-like functions, enabling explanation, extrapolation, and high flexibility. These parameter values are then used as the prior distribution. Another approach, proposed by Nalisnick et al. [82], implicitly models the prior distribution by controlling the complexity of the model, defined as the predictive divergence from a reference model. However, the use of the change of variables formula to translate this distance to parameter space is not elaborated upon here.

2.4 Optimization-based Approximate Inference

In optimization-based approximate inference, the main goal is to find the approximate distribution $q(w)$ that is closest to the true posterior distribution $p(w|D)$ via the optimization of some property that incorporate the distance between both. Typically $q(w)$ is a parametric distribution with parameters θ that belongs to a family of distributions $q \in \mathcal{Q}$. The choice of the approximate family $\mathcal{Q}$ is crucial to the overall approximation procedure. Practitioners often face a trade-off between choosing flexible approximate posterior to ensure that they are as closer to the true posterior and computational efficiency as at the same Q has to offer significant benefits. In the majority of cases we restrict the Q approximations so we can easily sample and evaluate their densities or to be able to compute easily expectations under this distribution. A significant amount of work proposes various directions for the approximate choice, ranging from simple Gaussians [41] to highly flexible implicit distributions [83]. We will discuss these further in Section2.4.3. Therefore, the main goal is to find the θ that best approximates the posterior distribution. This optimization is often defined via a cost function:

$$q_\theta(w) = \arg\min_\theta \; D(q_\theta(w)||p(w|D)) \tag{2.39}$$

Thus we aim to acquire the parameters θ that minimize the distance between the approximate q_θ and the true posterior distribution. Unfortunate this distance even for non deep learning models cannot be computed analytically. Way to circumvent the intractability of the optimization cost is to define a quantity that we can easily optimize and provides a guarantee of the overall optimization procedure. The most common choice of divergence measure $D(q_\theta(w)||p(w|D))$ to quantify the proximity between two densities is the information-theoretic Kullback-Leibler (KL). A lower bound based framework which is the most common is the Variational inference (VI). The goal is to acquire the parameters θ that minimize the distance between the approximate posterior distribution $q_\theta(W)$ and the true posterior distribution. Unfortunately, even for non-deep learning models, this distance cannot be computed analytically. To circumvent the intractability of the optimization cost, we define a quantity that can be easily optimized and provides a guarantee for the overall optimization procedure. The most common choice for this divergence measure, denoted by D, is the information-theoretic KL divergence. A common framework based on a lower bound is Variational Inference.

In this section, we will analyze optimization-based approximate inference, focusing particularly on the VI framework. This framework plays a crucial role in applications introduced in this dissertation. We will begin by introducing the basic form of the VI framework (Section 2.4.1). In Section 2.4.2, we will discuss methods and techniques for extending typical VI to deep learning models. Finally, Section 2.4.3 will explore several ways to utilize the VI framework for highly flexible posterior distribution approximations in deep neural networks.

2.4.1 Variational Inference

The VI optimization method implicitly optimizes Eq. 2.39 by optimizing a lower bound of the evidence function Eq. 2.11. While this connection might not be immediately apparent, the most appealing characteristic of the VI framework is its simplicity and tractability in most cases. We begin our discussion by introducing a tractable parameterization of the distance between the approximate posterior and the true posterior distribution.

$$\begin{aligned} D_{KL}(q(w)||p(w|D)) &= -\int q(w) \log \frac{p(w|D)}{q(w)} dw \\ &= -\int q(w) \log \frac{p(D|w)p(w)}{p(D)q(w)} dw \\ &= -\int q(w) \left(\log \frac{p(D|w)p(w)}{q(w)} - \log p(D) \right) dw \\ &= -\int q(w) \left(\log \frac{p(D,w)}{q(w)} \right) dw + \log p(D) \\ &= -\mathcal{L}(q(w)) + \log p(D) \end{aligned} \tag{2.40}$$

Where $\mathcal{L}(q(w))$ is called the *Evidence Lower Bound* (ELBO). This reparameterization allows us to easily optimize the KL-divergence with respect to θ, by directly optimizing the ELBO; which is a tractable quantity. The evidence $p(D)$ which is unknown, does not depend on the approximate distribution parameters θ and can therefore be treated as a constant during optimization. Thus minimizing the KL-diverge Eq. 2.39 is equivalent to maximizing the ELBO. To understand why the ELBO get its name

we can deploy its alternative derivation:

$$\begin{aligned}\log(p(D)) &= \int \log p(D,w)dw \\ &= \int \log\left(q(w)\frac{p(D,w)}{q(w)}\right)dw \\ &\geq \int q(w)\log\left(\frac{p(D,w)}{q(w)}\right)dw \\ &= \mathcal{L}(q(w))\end{aligned} \tag{2.41}$$

where Jensen's inequality takes advantage of logarithm's is a concave nature. Formally we defined the lower bound according to parameterization.

$$\mathcal{L}(q(w)) = \mathbb{E}_{q(w)}\log p(D|w) - D_{KL}(q(w)||p(w))dw. \tag{2.42}$$

Upon examining this equation, we can see that the ELBO is clearly composed of two terms. These terms form a relationship which allows different interpretations to emerge, each offering a distinct perspective on the overall optimization problem. From a statistical perspective, the ELBO reflects the trade-off between model accuracy and regularization penalty. A Bayesian point of view states that the ELBO reflects the balance between the data likelihood and the prior distribution over model parameters. Finally an information-theoretical approach states that there exists a strong connection between ELBO and the minimum description length principle, as described in [84].

Further analyzing Eq. 2.42, the KL divergence term can be interpreted as a regularizer that is minimized when the posterior distribution exactly matches the prior distribution. This definition of the ELBO has been used to explain the overregularization phenomenon observed in variational inference methods, where they tend to prune many weight parameters and lead networks trained with VI to sub-optimal performance (see Section 2.4.2). Therefore, the KL-divergence term's value ideally should be small. However, we don't want it to approach zero, as this prevents the variational posterior from capturing any meaningful dependencies in the underlying training data. Additionally the ELBO function can be also parameterized as follows:

$$\mathcal{L}(q(w)) = \mathbb{E}_{q(w)}\log p(D,w) - \mathcal{H}(q(w))dw. \tag{2.43}$$

This alternative parameterization highlights that, a "good" posterior approximation should assign most of its probability mass to regions with high joint probability density. Additionally, it should maximize the entropy of $q(w)$. This viewpoint helps

us understand the key difference between variational Bayesian inference and the MAP approach. In MAP estimation, the goal is to find a single value of w that maximizes the joint density, even if it resides in a region with very low posterior mass. In contrast, the entropy term Eq. 2.43 discourages $q(w)$ from collapsing into a single point [85].

2.4.2 Stochastic Variational Inference & Challenges

The ELBO optimization allows us to learn the parameters θ of the approximate posterior distribution, $q_\theta(w)$, using simple stochastic optimization. This enables the VI framework to scale to modern deep learning architectures. Therefore, stochastic variational inference (SVI) addresses this problem in the spirit of stochastic gradient descent. In each iteration, a random minibatch of size M is selected to obtain a stochastic estimate of the ELBO:

$$\mathcal{L}(q(w)) = \frac{N}{M} \sum_{m=1}^{M} \mathbb{E}_{q(w)} \log p(y_m|x_m, w) - D_{KL}(q(w)|p(w)). \tag{2.44}$$

Where m is defined as the variable index from the mini-batch. The gradient is then computed, providing a noisy estimate of the true ELBO's direction of steepest ascent. Stochastic variational inference shares the same convergence requirements as regular stochastic gradient descent. The minibatch indices, denoted by m, must be drawn uniformly at random. The minibatch size is denoted by M. Larger M values reduce the stochastic gradient's variance. When $M = N$ (dataset size), SVI reduces to traditional VI. To perform stochastic optimization, a tractable form for the ELBO gradients is needed:

$$\nabla_\theta \mathcal{L}(q(w), \theta) = \nabla_\theta \int q_\theta(w) \log \frac{p(D, w)}{q_\theta(w)} dw. \tag{2.45}$$

However, calculating the ELBO and its gradient often intractable as it involves the computation of the expectation w.r.t approximate posterior distribution. A common approach is to construct a Monte Carlo estimator for the ELBO and its gradient with respect to θ. The chosen estimator's properties, particularly its bias and variance, are crucial for the viability and accuracy of the stochastic optimization process. We will discuss two popular gradient estimators used to approximate the ELBO gradients in the following section.

Unbiased Gradient Estimators

First we consider the score function estimator also known as a likelihood ratio estimator or Reinforce, [86].

$$\begin{aligned}
\nabla_\theta \mathcal{L}(q(w), \theta) &= \int \nabla_\theta [q_\theta(w)] \log p(D, w) - \log q_\theta(w) dw \\
&+ \int \nabla_\theta [\log p(D, w) - \log q_\theta(w)] q_\theta(w) dw \\
&= -\mathbb{E}_{q(w)} \log q_\theta(w) \\
&+ \int \nabla_\theta [q_\theta(w)] \log p(D, w) - \log q_\theta(w) dw \\
&= \int \nabla_\theta \log q_\theta(w) q_\theta(w) (\log p(D, w) - \log q_\theta(w)) dw \\
&= \mathbb{E}_{q(w)} [\nabla_\theta \log q_\theta(w) (\log p(D, w) - \log q_\theta(w))]
\end{aligned} \tag{2.46}$$

In the above derivation the identity that, the expected value of the score function is equal to zero $\mathbb{E}_{q(w)}[\nabla_\theta \log q_\theta(w)] = 0$ and the log derivative trick which is related to the chain rule $\nabla_\theta[q_\theta(w)] = \nabla_\theta[\log q_\theta(w)] q_\theta(w)$, was used. The score function estimator depends only on the variational distribution, and not on the underlying model. Thus it can be trivially handle various variational approximations. Furthermore there are no assumptions about the form of the model (in fact non-differentiable models can works as well), only that the logarithm of the joint $p(D, w)$ can be computed, this significantly reduces the effort needed to implement variational inference in a wide variety of models [87].

The score function estimator, despite its benefits, also has drawbacks. It has been showed that stochastic gradients can exhibit high variance. The variance introduced by the Monte Carlo estimate can be too large for practical use and prevent the stochastic optimization process from converging. Additionally, high variance gradients necessitate very small optimization steps, leading to slow convergence. Therefore, variance reduction techniques are often employed when using the score function estimator. Common methods include Rao-Blackwellization, which reduces the variance of a random variable by replacing it with its conditional expectation with respect to a subset of variables [88], and control variates, a method where a stochastic term is added to the stochastic gradient such that its expectation remains the same, but its variance is reduced [89].

We will continue our discussion by deriving the pathwise gradient estimator, also known as the reparameterization trick (RT). While not as broadly applicable as the

score function estimator, the RT generally exhibits lower variance [90, 91]. It is applicable to continuous random variables whose probability density can be reparameterized. This reparameterization allows us to rewrite expectations. In the more general the goal is to compute the expectation w.r.t. $q_\theta(w)$ of a function f that depends on θ. The main goal of the pathwise estimator is apply change of variable and transform the expectation w.r.t $q_\theta(w)$ to an expectation of $q_0(\epsilon)$

$$\mathbb{E}_{q_\theta(w)}[f_\theta(w)] = \mathbb{E}_{q_0(\epsilon)}[f_\theta(\mathcal{T}(\epsilon;\theta))]. \tag{2.47}$$

In this setup q_0 is a fixed distribution with no dependence on θ and $\mathcal{T}(\epsilon;\theta)$ is a differentiable θ-dependent transformation. Since the expectation w.r.t. q_0 has no θ dependence, gradients w.r.t. θ can be computed by pushing ∇_θ through the expectation:

$$\begin{aligned}
\nabla\theta \int q(w)(f(w))dw &= \nabla\theta \int p(\epsilon)|\frac{d\epsilon}{dw}|f(w)dw \\
&= \nabla\theta \int p(\epsilon)|\frac{d\epsilon}{dw}|f(g(\epsilon,\theta))\frac{d\mathcal{T}(\epsilon,\theta)}{d\epsilon}dw \\
&= \nabla\theta \int p(\epsilon)f(\mathcal{T}(\epsilon,\theta))d\epsilon.
\end{aligned} \tag{2.48}$$

Where the change of variables formula was used for the density function which states that $q(w) = p(\epsilon)|\frac{d\epsilon}{dw}|$ and the change of variables theorem for integration $dw = \frac{dg(\epsilon,\theta)}{d\epsilon}d\epsilon$. Equality in the above equation holds for specific types of variational distributions. This reparameterization can be done for a number of distributions, including for example the Normal distribution. However, applying it to other commonly used distributions, such as the Gamma and Beta distributions, is non-trivial. This is because the required shape transformations $\mathcal{T}(\epsilon,\theta)$ often involves special functions [92].

As discussed, Monte Carlo gradient estimators exhibit high variance, especially when the gradient dimensionality extends to high-dimensional settings in Bayesian deep learning. Several attempts have been made to reduce variance in gradients during optimization. We will discuss two of the most widely used methods in variational deep learning.

In the case of stochastic ELBO optimization, the variance of the likelihood term $\text{Var}[\log p(y_i|x_i,w)]$ is large. This arises from the introduction of gradient covariances because different mini-batch instances share the same weight sample. One solution to this problem involves decorrelating the individual likelihoods by sampling one set of weights w per mini-batch input example. However, this incurs a great computational cost. Therefore the authors in [93] proposed the *local reparameterization*

trick (LRT) which avoids the exhaustive per-example weight sampling by directly working in the space of pre-activations. Following the same ideas as variance propagation methods (see Section 2.3.2), the LRT analytically computes the induced statistics of the pre-activations. The core idea of LRT is that rather than sampling the Gaussian weights and then computing the resulting pre-activations, it samples the pre-activations from their implied Gaussian distribution directly. The linear layer formulation can be utilized to serve as a concrete example of resulting mean and variance of the pre-activations. Having $B \in \mathbb{R}^{B \times O}$ with activations $A \in \mathbb{R}^{B \times I}$ and layer weights $W \in \mathbb{R}^{I \times W}$ the moments for the pre-activations can be defined:

$$\begin{aligned} \mathbb{E}[B] &= A\ \mathbb{E}[W] \\ \text{Var}[B] &= A\ \text{Var}[W]\ A^T. \end{aligned} \tag{2.49}$$

If one ignores the correlations between activations which occur in (2.49) by taking the diagonal part of the implied variance matrix then each activation $b_{i,j}$ is independent and the Gaussian distribution reduces to:

$$q_\phi(b_{i,j}|A) = \mathcal{N}(\gamma_{m,j}, \delta_{m,j}) \quad \text{where} \quad \gamma_{m,j} = \sum_{i=1}^{K} \alpha_{m,i}\mu_{i,j} \quad \text{and} \quad \delta_{m,j} = \sum_{i=1}^{K} \alpha_{m,i}^2 \sigma_{i,j}^2. \tag{2.50}$$

By doing so, the global uncertainty in the weights is translated into a form of local uncertainty that is independent across examples and easier to sample.

While the LRT has been widely popular and used extensively in variational inference applied to deep neural networks, there may be cases where directly transferring the sampling procedure to the pre-activations might not be the optimal choice or even possible. For example, consider relatively small neural networks (e.g., ResNet-11) acting on large-scale input images. In such cases, the dimensionality of activations is larger than that of the weights themselves. Consequently, while weights can be easily sampled from the approximate posterior distribution, sampling from the implied pre-activation distribution becomes computationally more expensive. Additionally, for layer operations that do not allow for easy analytical derivation of the full pre-activation distribution (e.g., squeeze-excitation blocks [94] , depthwise convolutions [95]), the LRT approach might not be feasible.

To address the variance problem in situation where the sampling is performed directly on weights, the authors in [96] proposed the *Flipout* method. This method shares a similar concept with pathwise reparameterization gradients. As the sampling from the main approximate distribution can be performed by transforming samples

coming from a base distribution typically (location-scale Gaussians). Assuming that the base distribution is symmetric around zero and samples are i.i.d. the authors propose to decorralate batch weights using pseudo-random samples. Specifically they profit on the property that $\hat{\Delta w} \sim q_\theta(w)$ sample from the approximate posterior identically distributed with the $\hat{\Delta w} \odot E$ where E is a random sign matrix. Flipout exploits this fact by using a base weight sample shared by all examples in the mini-batch, and multiplies it by a different rank-one sign matrix for each example:

$$\Delta w_n = \hat{\Delta w} \odot r_n s_n^T \tag{2.51}$$

where the subscript n denote the index within the mini-batch. The r_n and s_n vectors are sampled uniformly from ± 1. As noted in [96], the marginal distribution over gradients computed for individual training examples is identical to the distribution computed using shared weight perturbations. This procedure can be formulated in matrix form, making the sampling process quite efficient in practice. Furthermore, it leads to an estimator with low variance and a model with good empirical performance.

Overregularization & Cold Posteriors

Directly optimizing the ELBO is a challenging task. As we discussed, the two terms composing the ELBO the likelihood and the prior are often difficult for practitioners to balance effectively. Even in the pioneering work of applying variational inference to neural network parameters [84], some form of heuristic manipulation had to be applied to the ELBO in order to derive a more stable optimization procedure. The most common manipulation involves scaling the KL divergence term by a small factor, denoted by λ:

$$\mathcal{L}(q(w)) = \mathbb{E}_{q(w)} \Big[\sum_{m=1}^{N} \log p(y_m | x_m, w) \Big] - \lambda D_{KL}(q(w) | p(w)). \tag{2.52}$$

This technique has been quite useful in providing models with good empirical performance, and it has been common practice even in modern applications of VI [97][98]. The KL term pushes the posterior distribution closer to the prior (which is often a simpler distribution). This is the case when sparse priors are deployed (see Section 2.3.3), which in some cases can lead the weights to become inactive during training, effectively reducing model capabilities. This can have a strong effect on a large number of weights, often pushing their values towards local optima, which can

result in significant underfitting of the data. Sonderby et al. [99] proposed a dynamic λ factor that starts from zero, allowing the model to fit the data initially. This factor gradually increases to one, promoting sparsity as training progresses. This approach is particularly beneficial when there is a significant discrepancy between the number of weights and the number of training samples.

The authors [100] made a connection that annealing the KL term in the ELBO is connected with the famous *cold posterior effect*. According to this when performing inference in Bayesian models, the posterior can be tempered by a positive temperature factor T:

$$\log p(w|D) = \frac{1}{T}\left[\log p(D|w) + \log p(w)\right] + C. \tag{2.53}$$

Temperature values $T < 1$ (i.e., cooling the posterior) have been found to achieve better predictive performance. This downweighting of the KL term corresponds to a different ELBO that minimizes the divergence between the approximate posterior $q_\theta(w)$ and a partial cold posterior, where only the likelihood term is tempered with $T < 1$. A version of the ELBO that directly minimizes the divergence with respect to the full cold posterior was derived in [101]. This work showed that well-tempered posteriors lead to even better predictive performance and improved uncertainty calibration in computer vision tasks. There are several hypotheses that attempt to explain the cold posterior phenomenon. One such hypothesis is the 'data augmentation hypothesis' proposed in [100]. This suggests that suppressing data augmentation might be sufficient to remove the cold posterior effect. Alternatively, the cold posterior effect may simply be an phenomenon of a misspecified prior or inaccurate inference. While the cause of posterior scaling and its connection to the "cold posterior effect" remain active research topics, there is debate among researchers regarding the impact of these techniques on Bayesian principles. Some argue they invalidate these principles, while others propose alternative explanations and solutions [102].

2.4.3 Beyond Gaussian Approximate Distributions

In the previous sections, we have focused on cases where the approximate posterior remains a Gaussian-like distribution. Although these distributions offer desirable mathematical properties that simplify the ELBO calculation and often lead to analytical computation of the KL divergence, restricting the variational posterior distribution to the Gaussian family can be problematic. This is because the Gaussian distribution

may not be flexible enough to capture the potentially complex latent space of the weight variables. Intuitively, more complex variational distributions should theoretically reduce the distance to the true underlying posterior, thereby yielding better results. The fact simple approximate weight posteriors often fail to outperform non-Bayesian posterior (like Deep Ensembles) and especially in the case of covariance shift they seem to under perform, might indicate that the posterior might have multimodal shape, as the example in Figure 2.5 indicates. While BNNs have previously been shown to provide uncertainty estimates that are useful for a range of tasks, authors in [103] theoretically proved that VI with diagonal covariance Gaussian approximate posterior (also reffed to as mean-field) has an additional gap in the evidence lower bound compared to a purpose-built posterior. Many works try to explain this behavior by exposing the problems of simple mean field variational inference. It was empirically found in [104], that the maximization of the ELBO function in the mean field setting is prone to yield very poor inferences.

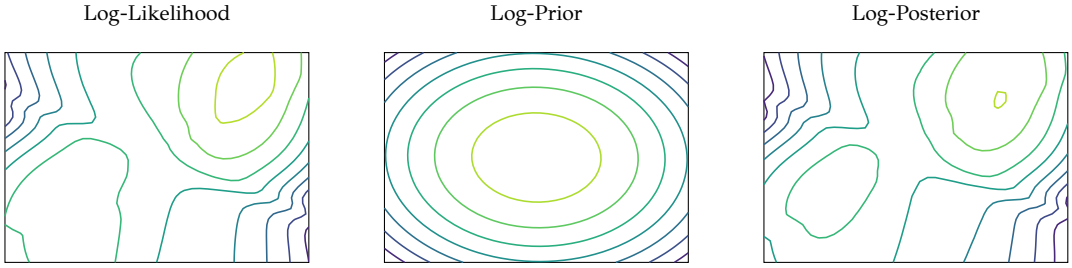

Figure 2.5: Posterior density visualization for a small MLP on 1D regression. Visualization of posterior log-posterior, log-likelihood and log-prior in the two- dimensional subspace of the parameter space spanned by three HMC samples (Inspired from [38]). We can observe that even this simple NN can induce multimodal posterior attributes.

Several approaches exist to model neural network weights with distributions more flexible than Gaussians. A simple example involves introducing an auxiliary variable, augmenting the weight distribution in an hierarchical way (see Section 2.3.3 and Figure 2.4). While these methods achieve richer distributions, they are still restricted to unimodal shaped posteriors. The remainder of this section focuses on two methods for approximating posterior distributions with minimal restrictions on the target distribution's form. Both methods leverage a network to transform samples drawn from a base distribution into an approximation of the target approximate weight posterior. These network that are used to predict deterministically the parameters of another,

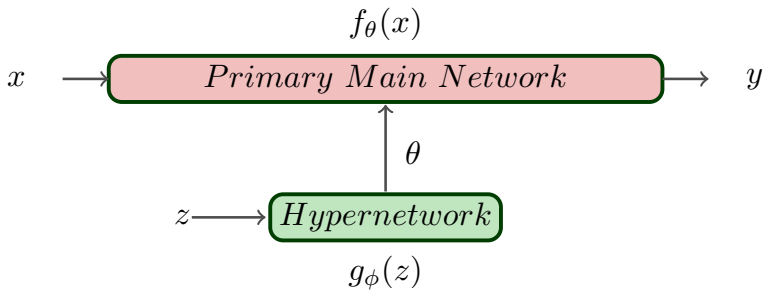

Figure 2.6: This figure illustrates the hypernetwork learning framework. A smaller hypernetwork, $g_\phi(z)$, with parameters ϕ takes an input, z, and generates parameters, θ, that influence the parameters of the larger main neural network, $f_\theta(x)$. Both networks can typically be learned using backpropagation from the final loss on the predicted output, y.

typically larger network, termed the *primary* network [105] are called *hypernetworks* and where extend to a Bayesian setting [106].

Normalizing Flows

Normalizing flows are a flexible approach for defining complex probability distributions over high-dimensional data. They achieve this by transforming a samples from a base distribution (e.g., a standard normal distribution) into samples generated by a more complex and flexible one, while preserving valuable mathematical tractabilities.

The core idea of flow-based modeling is to express a new variable w' as a transformation f applied to a real vector w sampled from the base distribution $p(w)$,

$$w' = f(w) \quad \text{where} \quad w \sim p(w). \tag{2.54}$$

The defining property of flow-based models is that the transformation f must be invertible and both f and f^{-1} must be differentiable. Such transformations are known as diffeomorphisms and require both w' and w to have the same number of dimensions [107]. The density of w' can be obtained via:

$$q(w') = q(w) \left| \det \frac{\partial f^{-1}}{\partial w'} \right| = q(w) \left| \det \frac{\partial f}{\partial w} \right|^{-1}. \tag{2.55}$$

Where the last equality holds by applying the chain rule and is a property of Jacobians of invertible functions. Arbitrarily complex densities can be constructed, by composing

several simple maps and repetitively applying (2.55). After K sequential mapping applied on the initial random variable w_0 then the density $q_k(w_k)$ is obtained by:

$$\log q_k(w_k) = \log q_0(w_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial w_{k-1}} \right|. \tag{2.56}$$

The path which is modeled by the successive distributions q_k is a normalizing flow. A property of such transformations, often referred to as the law of the unconscious statistician (LOTUS), is that expectations w.r.t. the transformed density q_k can be computed without explicitly defining it. Any expectation $\mathbb{E}_{q_k}[h(w)]$ can be written as an expectation under q_0 as:

$$\mathbb{E}_{q_k}[h(w)] = \mathbb{E}_{q_0}[h(f_k \circ f_{k-1} \circ \cdots \circ f_1(w))]. \tag{2.57}$$

As an illustrative example of a diffeomorphisms transformation is the ResNet-like transformation proposed by Rezende et al. [108] which is defined as follows:

$$f_k(z_{k-1}) = z_{k-1} + w_1 h(w_2^T z_{k-1} + b). \tag{2.58}$$

Where the vectors w_1, w_2 and the scalar b are the learnable parameters of the transformation and $h()$ is a non linear function. The term $w_1 h(w_2^T z_{k-1} + b)$ can be interpreted as an MLP with a hidden layer with a single unit. This transformation if it is stacked to flow has been proven to be able to capture high-dimensional dependencies. However it has significant limitations as its inverse cannot be computed easily. Affine transformations are also an option, but their expressivity is limited. To address this, the authors in [109] introduced a coupling method that enables highly expressive transformations for flows. Similarly, [110] utilized autoregressive models as a form of normalizing flow. For a comprehensive review of available transformations, please refer to [111].

Normalizing flows can be used in variational inference to model the approximate posterior distribution $q_\phi(w|D) = q_k(w_k)$, and the ELBO takes the following form:

$$\begin{aligned}
\mathcal{L}(q(w), \theta, \phi) &= \mathbb{E}_{q_\phi(w|D)} \left[\log \frac{p(D, w)}{q_\phi(w|D)} \right] \\
&= \mathbb{E}_{q_0(w_0)} \left[\log q_K(w_K) - \log p(D, w_K) \right] \\
&= \mathbb{E}_{q_0(w_0)} \left[\log q_0(w_0) \right] - \mathbb{E}_{q_0(w_0)} [\log p(D, w_K)] - \mathbb{E}_{q_0(w_0)} \left[\sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial w_{k-1}} \right| \right].
\end{aligned} \tag{2.59}$$

Normalizing flows can be used with any variational optimization scheme. Direct application of Bayesian deep learning in the framework of variational inference is not straightforward. The computation of the ELBO, necessitates flows that are efficient to sample from and their densities can be evaluated easily. These requirements inherently limit the scalability of normalizing flows for directly modeling the weights of a deep neural network. However there are ways to address this issue. A popular way is to restrict flows to model a certain part of the model weights. As shown in [112], where flows can be used to model specific components of the weight distribution. Furthermore the weights can be mapped to a lower-dimensional space [113], where normalizing flows become more computationally efficient.

Implicit Distributions

Another key approach for modeling flexible approximate posterior distributions comes in the form of implicit distributions. These distributions are implicitly defined through their generative process. This allows sampling from these distributions, but neither their density nor their gradients of their density can be evaluated, hence the term "implicit". Similar to normalizing flows, they can be defined define a transformation function, which typically takes the form of a network, to transform/generate samples from a base distribution. In the case of modeling neural network weights, this parallel transformation network generates weight samples. However, unlike normalizing flows where the transformation network does not need to be invertible, the involvement of implicit distributions makes the probabilistic modeling more general and significantly more scalable to modern deep neural networks.

The generative process involves two distributions: $z \sim q(z)$ and $w \sim q_\theta(z)$. Here, $q(z)$ is a fixed base distribution, while $q_\theta : \mathbb{R}^d \to \mathbb{R}^m$ is a non-linear, typically non-invertible mapping. This non-invertibility makes it challenging to directly compute the density function. Although the likelihood term in the ELBO Eq. 2.45 alongside its gradients can be estimated using Monte Carlo methods and the reparameterization trick [90]. The same is not true for the regularization term which involves the computation of $D_{KL}(q_\theta(w)||p(w))$, as a closed-form solution is unavailable. Furthermore, $q(w)$ is not a well-defined density in the parameter space. Its support lies on a low-dimensional manifold and has measure zero, leading to potential issues with the KL divergence not being well-defined.

Several approaches exist to efficiently and stably approximate the KL-term for

implicit distributions. Works like [83] found that a single discriminator network is inaccurate at estimating log-ratios proposing an analytical form by deploying a kernel method. A more recent approach further decomposes the KL-term in the ELBO. It first proposes and tests an entropy approximation for an implicit density. They use a linearization method to compute the integral $\mathbb{H}[q_\theta(w)]$. This method considers an analytical approximation (see Sec. 2.3.2) obtained via a local linearization of the generator/neural sampler around z. This approach, along with some modeling choices, provides an analytical approximation to the implicit-VI ELBO, leading to significant performance boost.

2.5 Efficient Approximate Bayesian Inference

Although their success in providing well-calibrated predictions for medium-sized neural networks, Bayesian inference methods discussed in this chapter often struggle to scale effectively to deep neural networks. An example of a method with moderate scalability is the Mean-Field Variational Inference (MFVI). Despite its foundational role in constructing well-known models (e.g., Variational Mixture of Gaussians, Bayesian Relevance Vector Machine) even before the deep learning era, suffers from limitations in the posterior approximation constraint. This limitation becomes particularly significant when the dimensionality of the variational posterior distribution grows, leading to sub-optimal performance in deep networks. Consequently, achieving a fully Bayesian treatment that delivers excellent performance in modern deep networks remains a challenge.

In practice, the weight size limits the posterior's expressiveness. Consider the Laplace approximation, for example. As the dimensionality of the weights increases significantly, the only practical option for covariance parameterization becomes a restrictive diagonal or Kronecker factorization. While this approach performs well in practice, it penalizes the ability to capture correlations. The same limitation applies to other methods that use a Gaussians to approximate the weights, hindering their ability to capture correlations in the approximate posterior. Another crucial aspect when dealing with deep neural networks is the overall training cost. This is particularly true for foundational models with the introduction of large language models (LLMs) like BERT [114] or ViTs [115]. Even retraining or, in some cases, fine-tuning these

transformer-based models becomes computationally prohibitive in many scenarios. Consequently, traditional optimization-based approaches for probabilistic reasoning must be adapted to address these challenges. Therefore, practitioners strive to introduce probabilistic treatment to deep neural networks efficiently by leveraging available information.

2.5.1 Advances in Scalable Bayesian Inference

This section explores methods of fusing Bayesian inference with deep learning models. We will distinguish some key methods for applying low-dimensional Bayesian deep learning. It's important to acknowledge that numerous approaches exist for probabilistic reasoning within deep neural networks.

One way to scale Bayesian inference is to consider them as models where a subset of the parameters are treated as random variables, while the remaining parameters function deterministically. A common and straightforward approach involves last-layer BNNs. These models only assign probabilistic treatment to the final layer, resulting in a linear model. This choice ensures analytical tractability for both inference and predictive distribution, similar to Gaussian processes. Additionally, the remaining neural network structure acts as a feature extractor [116, 117, 118, 119]. Following the same notion researchers in [120] proposed a method to select a subset of weights that minimizes the discrepancy between the full posterior and an approximation formed by a smaller subset. This approach allows for high-fidelity inference on the smaller subset, leading to good results. In fact, the work in [121] even questions the necessity of a fully Bayesian treatment for large BNNs. Their findings suggest that accounting for the complete posterior doesn't necessarily lead to better performance.

Parallel to the aforementioned approaches, authors in [122] proposed to learn a distribution over non-linear projections of the weights, finding a latent representation for the weights directly rather than projecting the nodes. This involves projecting the weight space and applying inference there, essentially learning a non-linear latent representation of the network weights. This approach builds upon the work of [113], who identified a low-dimensional subspace of neural network weights. Inference is then applied directly on this subspace of weights, promoting more flexible approximate posteriors due to the drastically reduced dimensionality.

Another line of work that can scale to large neural networks involves introducing

stochastic noise to the network weights. These approaches typically utilize a multiplicative low-dimensional stochastic variable whose primary purpose is to induce stochasticity in the weights. The low dimensionality allows the noise to originate from highly flexible and multimodal posterior distributions, theoretically making the distribution of the whole network non-Gaussian. Louizos et al. [112] employed low-dimensional noise derived from a normalizing flow trained using variational inference (VI), while Dusenberry et al.[123], inspired by [124], used rank-1 multiplicative noise components sampled from a mixture of Gaussian distributions to enable multimodal posteriors.

As applications and model architectures become increasingly complex, research has shifted towards promoting model-agnostic uncertainty estimation strategies. These approaches introduce stochasticity by adding external modules to pre-defined and trained models. For example, Deng et al. [125] propose BayesAdapter, a method that performs post-hoc ELBO optimization of the weights. Similarly, Franchi et al. [6] introduce techniques inspired by convolutional variational neural pruning [126]. Building on these ideas, we propose a novel Bayesian normalization layer by modifying existing normalization layers with a Gaussian perturbation.

2.6 Discussion

This chapter lays the foundation for the remainder of the thesis. Here, we defined how deep neural networks are constructed and highlight their probabilistic modeling. We then discussed how each element of Bayesian inference is addressed appropriately in the high dimensionality of neural networks. Building on the concepts introduced here, subsequent chapters will explore how Bayesian treatment can be applied to vision models. Specifically, Chapter 3 will discuss the creation of robust and flexible vision object detectors using probabilistic feature fusion. Chapter 4 will combine the hypernetwork modeling from Section 2.4.3 with flexible generative models to introduce low-dimensional Bayesian inference for neural networks. Finally, Chapter 5 will elaborate on modeling weight correlations in the approximate posterior distribution. We will motivate their usage, highlight their benefits, and outline a covariance parameterization that can efficiently model weight correlations.

Chapter 3

Probabilistic Object Detection via Variational Feature Pyramid Networks

---

This chapter focuses on probabilistic object detection, specifically how to develop robust and well-calibrated detector models that provide reliable uncertainty estimates for each predicted object within an image. We propose a novel approach that reinterprets fusion networks, a core component of most detection deep learning architectures, with a sparse probabilistic architecture. To ensure efficiency, we train the network to identify the most relevant components specific to the task and dataset at hand. This approach achieves excellent accuracy in our numerical experiments.

The rest of this chapter is structured as follows. We begin by introducing the challenges of object detection with deep neural networks, in Section 3.1. We explore ways of how a probabilistic approach to detectors can alleviate them and briefly highlight our contribution. Section 3.2 focuses into the fundamental related work that significantly influences our contribution. This includes research on future fusion

network, stochastic neural network architectures and probabilistic deep learning detection methods in general. Section 3.3 provides a detailed review of our proposed fusion network named Variational Feature Network (VarFPN). We discuss the design choices we made and our Bayesian interpretation in great extend. The experimental evaluation of our method, combined with various detectors for both detection and segmentation tasks, is presented in Section 3.4. We numerically evaluate our method's predictive performance in object detection and instance segmentation tasks, followed by an evaluation of the uncertainty estimates. Furthermore, we emphasize the probabilistic benefits of the proposed stochastic architecture modeling in a low training data semantic segmentation task. The chapter is concluded in Section 3.5 with a discussion on our contribution.

3.1 Challenges of Object Detection & Contributions

Object detection and instance segmentation are two of the most fundamental problems in the computer vision field. These problems are quite difficult to solve and provide multiple challenges as multiple objects have to be detected or segmented at multiple scales, locations and under different conditions. The majority of those problems in practice are today approached via the use of deep learning architectures. In the recent years multiple deep architectures have been proposed, leading to widely known models in computer vision [127, 128]. Neural network-based detectors are typically built and designed upon deep robust feature extraction (sub-)networks referred to as backbones. The task of these components is to transform the input image to a deep embedded representation, subsequently fed to the detector head in order to have it produce the required predictions.

These networks heavily rely on good representations, as the input feature must be descriptive enough to capture the different relations and scales among the image objects. A common practice is to use pretrained deep convolutional backbones such as ResNet [20] or Inception [129] to first extract rich features at different scales from the input image. Subsequently, their output is processed using a pyramidal-shaped multi-scale fusion network in order to create richer and more descriptive features, see Figure 3.1 for a graphical illustration. Since the introduction of Feature Pyramid Networks (FPNs) [1], which standardize the above procedure, works have been proposed

to design more flexible, sophisticated and also efficient fusion networks pushing even more the accuracy boundary limits of deep detectors.

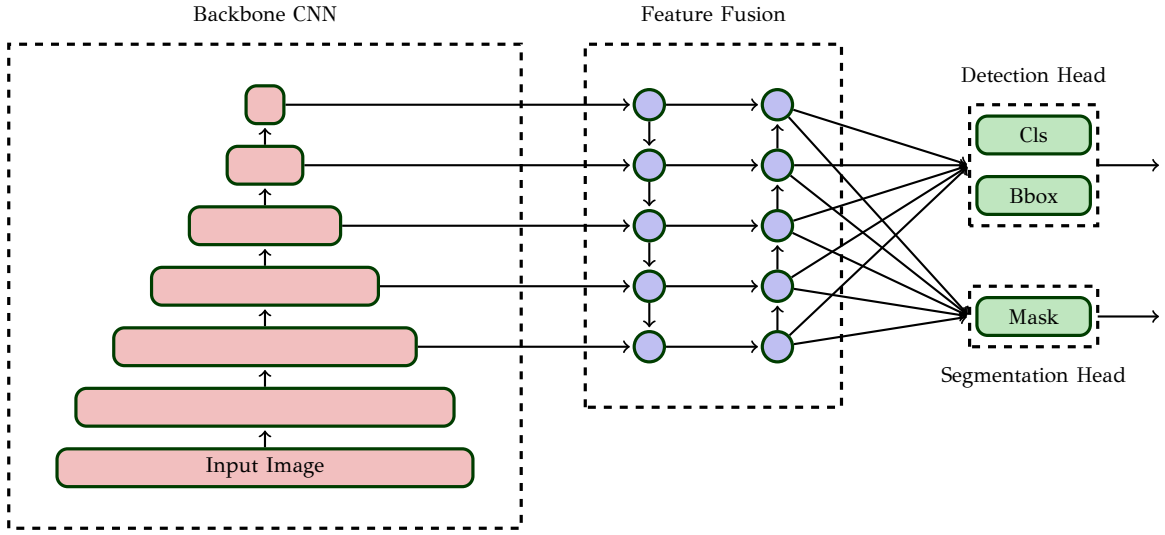

Figure 3.1: A high level object detection/segmentation architecture, which is based on CNN features.

While the accuracy of object detectors is crucial for safety-critical systems like autonomous driving, it's not the only factor. Robustness aspects, such as the ability to estimate uncertainty at the detection level, are equally important but often overlooked. Our work addresses this gap by leveraging Variational Inference to create sophisticated feature fusion networks for object detection tasks. This approach is applicable to various deep learning-based detection frameworks. Most common detection models share a similar high-level architecture consisting of three modules: a backbone network for feature extraction, a fusion network to combine features at different scales, and a head component responsible for making detections. We focus on the fusion network sub-module and propose a novel design with a distinct characteristic: a sparse probabilistic architecture. Our model leverages the ability to automatically adapt its architecture based on the training dataset's characteristics. This allows our model to achieve superior results while maintaining computational efficiency comparable to other related methods. Beyond promising performance, our approach based on probabilistic modeling enables any detection network to be easily transformed into a BNN, allowing it to predict uncertainty estimates for its outputs. Although our method only incorporates probabilistic modeling of a small number of parameters (around 100), the crucial role of feature fusion in overall detector performance allows us to reap

the benefits of a proper probabilistic model treatment.

3.2 Related Work

This section reviews related work on sophisticated feature fusion modules for object detection. We explore methods that propose and design these modules to provide richer and more robust features for task-specific heads. Additionally, we examine methods that introduce stochasticity into neural network models through probabilistic pruning of components, leading to stochastic architectures. Finally, we consider approaches for incorporating uncertainty estimation into detection methods and models.

3.2.1 Feature Fusion Networks

Feature fusion networks in the object detection framework was first proposed with the introduction of Feature Pyramid Networks (FPN) [1]. These networks offered an architectural solution to providing a multi-scale feature representation of the input. A "pyramidal"-structured hierarchy of feature maps is to be produced by a convolutional pipeline and combined to high-level semantic outputs. A bottom-up and a top-down pathway are the basic structural sub-elements of the feature pyramid. The bottom-up pathway computes a feature hierarchy, where each level corresponds to a different resolution scale. On the top-down pathway, starting from the coarsest resolution towards the finest one, feature maps are progressively upsampled (e.g. by a constant factor of 2), and combined with corresponding bottom-up maps. Over each pair of top-down and bottom-up corresponding blocks, the top-down map is semantically high-level, and the bottom-up map is semantically low-level.

There are a lot of recent works that propose sophisticated modules to extract more representative features for object detection tasks. Telling examples include [130], where the Pconv module is introduced to simultaneously extract features at different scales; attention-based modules [131, 132]; or even methods that discard the whole FPN structure [133] and methods that incorporate the popular Transformer architecture [134]. PANet [135] adds an extra bottom-up pathway on top of the original FPN architecture. The M2Det object detector [136] extends the idea and builds stronger feature pyramid representations by employing multiple U-shape modules after back-

bone pipeline.

Another approach, more related to our method, is NAS-FPN [137]. This approach uses a Neural Architecture Search (NAS) algorithm to find an optimal structure instead of manually designing architectures for pyramidal representations. This model requires a significant computational load for training, and the output network is irregular and difficult to interpret or modify. In [5], the proposed BiFPN model uses less building blocks as the authors drop blocks with only one input feature map. Furthermore, they add an extra edge linking the original input to an output node if they are at the same level, in order to fuse more features while avoiding too much extra cost. Another novelty of BiFPN is weighted fusion, with which they introduce a set of learnable parameters associated with each input feature map on every block. In this manner, the network is allowed to learn the importance of each separate feature map.

3.2.2 Probabilistic Pruning & Stochastic Architectures

Several works address the limitations of deterministic models by incorporating stochastic modules and architectures into their design [138]. This approach offers two main benefits. First, the inherent randomness in these architectures likely helps to regularize the training process. Second, the ”plug-and-play” nature of these models allows us to evaluate the same input data with a diverse range of architectures once training is complete. This not only reduces the training effort required to explore a broad spectrum of architectures, but it also leverages the unique inductive biases inherent to each architecture, potentially leading to more robust predictions. Furthermore, the predictions from these diverse architectures can be combined to estimate uncertainty, resulting in a more accurate, robust, and calibrated prediction model.

There are several works which adopt stochastic architectures. Works like [139] introduce DropBlock, where in the spirit of [140], is a form of structured dropout, where units in a contiguous region of a feature map are dropped together. Applying DropbBlock in skip connections in addition to the convolution layers increases the accuracy. The authors in [141] proposed a model where different depths correspond to subnetworks sharing weights. By marginalizing the predictions from these subnetworks, their approach captures model uncertainty.

In the spirit of [142], methods that deploy stochastic architectures often augment

a deterministic network with differentiable architecture parameters that have the ability to prune away several computational paths. Straightforwardly with this modeling choice, literature on sparse networks and sparse training is highly relevant. The notion of pruning parameters of deep neural networks comes to relax the implementation difficulties on resource-constrained platforms. In general a successful pruning method must be able to compress the model and improve efficiency with a minimal loss in terms of accuracy. To this end, probabilistic approaches using Variational Inference have already been deployed. [93] treat weights of neural networks as random variables and leverage Variational Inference to efficiently estimate the parameters, in a model that is shown to elegantly generalize Gaussian dropout [143]. [144] revised the previous work and proposed a scheme to estimate the dropout rate, proving that the resulting method leads to sparse solutions. Further works, imposing sparse priors on the weights [145], proposed the use of hierarchical priors on hidden units; on a different note, neurons can be pruned altogether, including all their incoming and outgoing weights. This avoids more complicated and inefficient encoding schemes. Unfortunately, these methods cannot be generalized to complex convolutional layers found in modern deep learning models due to the complexity and interdependence of operations. To tackle this problem, more general probabilistic methods of network pruning have been proposed. In [126] for example, the batch normalization layer is reformulated, where the normalized features are multiplied channel-wise with a sparse prior-based stochastic parameter that effectively prunes redundant channels.

3.2.3 Probabilistic Object Detection

Integrating Bayesian and probabilistic principles into object detection has long been a focus for researchers. Scene image understanding is inherently uncertain, with factors like unclear object boundaries and partial occlusion making object labels ambiguous. Probabilistic object detection addresses this challenge by detecting objects in images while simultaneously quantifying the spatial and semantic uncertainties associated with those detections [3]. Building upon these ideas the pioneering work by Kendall et al. [33], models both aleatoric and epistemic uncertainties for tasks like pixel-wise semantic segmentation and depth regression . The authors highlight the importance of epistemic uncertainty in safety-critical applications and for scenarios with limited training data.

Thus researchers have increasingly focused on incorporating epistemic uncertainty into their object detection models. Two popular approaches to introduce uncertainty are widely implemented [146]. First approach utilizes Monte Carlo Dropout (MC Dropout) [147]. Methods based on this approach leverage the existing dropout layers commonly used in object detectors, making it a straightforward way to achieve uncertainty estimation. MC dropout based methods [148] [149] estimate both spatial and classification uncertainties for object detection and use the uncertainty to accept or reject detections under open-set conditions. Frameworks such as BayeOD [150], also leverage MC dropout to approximate the posterior over parameters, while directly replacing standard non-maximum suppression (NMS) with Bayesian inference, This allows the detector to retain all predicted information for both the bounding box and the category of a detected object instance. The second approach is the Deep Ensembles method, which shares similar benefits with MC Dropout in terms of uncertainty estimation and ease of implementation. Works like [151] utilize deep ensembles to quantify the uncertainty in overall detection results. They propose a merging algorithm for Deep Ensembles that combines detections from various models based on the predicted object similarity. In contrast to these methods, works like [7] focus on estimating aleatoric uncertainty, which reflects inherent noise in the data. They directly estimate the spatial uncertainty as a Gaussian distribution and propose an uncertainty-aware NMS method for suppressing redundant predictions of the same object.

Several non-Bayesian methods aim to improve the calibration of object detection networks, enhancing their robustness to distribution shifts and data degradation. These methods achieve this by producing more calibrated predictions. For instance, authors in [152] propose a Gaussian process (GP) recalibration scheme that yields parametric output distributions. This approach addresses the issue of standard detection models overestimating spatial uncertainty. Similarly, in the context of object classification calibration, additional information from an object detector's regression output is leveraged for calibration [153]. Notably, this method can jointly calibrate multi-class confidence and box localization by exploiting their predictive uncertainties. Additionally, several non-probabilistic techniques have been explored, including self-distillation [154] and self-prediction ensemble techniques [155].

3.3 Variational Feature Pyramid Networks

We now formally introduce our proposed feature fusion network that combines aspects proposed in recent works such as [5], which we will use as a network baseline. This fusion network is composed by multiple computation nodes or building blocks with dense pathway connections between them. Each connection is augmented with an architecture weight. Subsequently, instead of using deterministic architecture weights, we show how to estimate the distribution of architecture weights cast as random variables via the use of variational inference. We analytically describe how the Stochastic Gradient Variational Bayes (SGVB) estimator [90] (See Section 2.4.2) can be used to apply inference on the fusion weights. Finally, we introduce sparse prior distributions like Automatic Relevance Determination (ARD) to effectively prune network connections, thus obtaining a model with the optimal lower complexity.

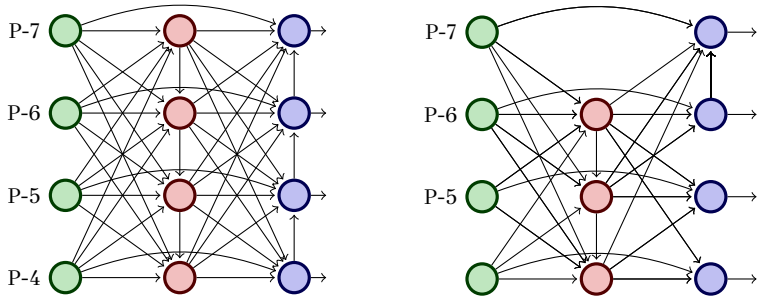

Figure 3.2: An illustration of pruning under the proposed method. The image on the left depicts the initial model before training which is highly complex, including multiple-level feature fusion and is "fully" connected. On the right, the same network is shown after 10 epochs of training, where redundant connections and building blocks have been pruned, leading to an efficient fusion network.

3.3.1 Proposed Feature Fusion Network

The architecture of our network is initialized as a 'fully connected' version of the PaNET [135] (with bottom-up and top-down pathways) using skip connections. On each building block, the input features are combined via the use of fast normalized fusion, an efficient approximation of softmax proposed in [5]. An illustration of this initial architecture is depicted in Figure 3.2 (left). The output of a building block F_{out} in our network is formally defined as follows. All building blocks are considered to

be identical. Let $F_{level}^{layer} = \{F_1, \ldots, F_N\}$ be the set of all N features that are input to the given block and $W_{level}^{layer} = \{w_1, \ldots, w_N\}$ a set of weights $w_i \geq 0$ each of which is associated with one input feature from F_{level}^{layer} [1]: Then for F_{out} we write:

$$F_{out} = \text{Conv}\left(\frac{\sum_{i=1}^{N} w_i F_i}{\sum_{i=1}^{N} w_i + \epsilon}\right), \tag{3.1}$$

where ϵ is a small constant added for numerical stability and Conv stands for a 2D convolutional operation. Note that all features in F_{level}^{layer} are resized using bilinear interpolation when they correspond to cross-scale connections. The use of normalized fusion to re-weight feature, such that all weights are normalized to be a probability with value range from zero to one. Representing the importance of each input stabilizes the training and allows smoother architecture pruning.

3.3.2 Variational Inference

Our method treats each weight w associated with a network connection as a stochastic variable drawn from a parametric distribution $p(W)$. This probabilistic approach, as we already discussed, offers two key benefits. First, it elegantly promotes sparsity, leading to lightweight fusion networks. Second, as the weights can amplify or suppress the importance of specific features within the fusion process, they play a crucial role. By treating these parameters probabilistically, we can achieve calibrated and robust features and acquire uncertainty estimates for the features most relevant to the specific dataset the detector network is trained on. We will study this argument deeper in in our experimental results.

Up to this point we consider a detection dataset $D = \{(x_j, y_j)\}_{j=1}^{J}$ as a set of random variables, where x is input data and y is the corresponding ground truth, in a dataset comprised of J images. The joint distribution which combines and depicts the relations of model random variables is defined in the following way:

$$p(X, Y, W) = p(Y|X, W)p(W). \tag{3.2}$$

Our goal is to estimate the posterior distribution of latent variables W i.e. $p(W|Y, X)$. Since the posterior distribution cannot be obtained in closed form, we cannot apply exact inference methods, thus we resort to approximate inference and specifically

[1] At this point, we want to make clear to the reader, that throughout this chapter notation w refers only to the architecture/connection weights.

to the variational Bayesian methodology (see Sec. 2.4.1). We assume a family of approximate posterior distributions $q_\phi(W)$ parameterized by ϕ, and then seek values for the parameters ϕ that best approximate the true posterior. The best approximation of the true posterior distribution comes via maximizing the lower bound ELBO, a process which as we showed in the previous chapter is equivalent to minimizing the KL divergence. Unfortunately, the required integrals for applying a mean-field VB algorithm are also intractable. These intractabilities appear since the detection networks are extremely complicated likelihood functions $p(Y|X,W)$. Thus, in order to find the posterior distribution w.r.t. hidden variables we directly optimize the ELBO, which we in this case can be written as:

$$\mathcal{L}(W) = \mathbb{E}_{q_\phi(W)}[\log p(Y|X,W)] - KL(q_\phi(W)||p(W)). \tag{3.3}$$

We want to differentiate and optimize the ELBO $\mathcal{L}(W)$ w.r.t. both the variational parameters ϕ and generative parameters θ. However, the gradient of the ELBO w.r.t. ϕ is not trivial to compute. We proceed by using pathwise gradient estimator/reparameteriarion trick (see Sec. 2.4.2) [90] and use an estimate $\tilde{\mathcal{L}}(W) \simeq \mathcal{L}(W)$ of the lower bound and its derivatives w.r.t. the parameters. According to RT, the approximate posterior $q_\phi(W)$ can be reparameterized via a differentiable transformation $f(\phi,\epsilon)$ of an (auxiliary) noise variable ϵ:

$$w = f(\phi,\epsilon), \quad \text{where} \quad \epsilon \sim p(\epsilon) \tag{3.4}$$

We can now form a low-variance Monte Carlo estimate on the expectation appearing in (3.3). Under the change-of-variables rule for integrals, the expected log-likelihood is the same as the expectation w.r.t. the auxiliary distribution

$$\tilde{\mathcal{L}}(W) = \frac{1}{L}\sum_{l=1}^{L}\sum_{j=1}^{J}\log p(y_j|x_j, W = f(w,\epsilon_{l,j})) - KL(q_\phi(W)||p(W)), \tag{3.5}$$

where $\epsilon_{l,j}$ is the l^{th} sample of $p(\epsilon)$ for the j^{th} input datum. The advantage of using the reparametrization trick is that we can construct Monte-Carlo estimators with low variance, leading to more efficient training in the context of use with a variational mode, as long as we can find functions $f(\phi,\epsilon)$ and fixed distribution $p(\epsilon)$ given some distribution $q_\phi(\cdot)$.

3.3.3 Choice of Prior Distribution

Here we explicitly define the prior distribution for the connection weights, the choice of the approximate variational posterior distribution and the ELBO. Regarding the type of prior distribution we have to account for several things. First, since we want the fusion architecture to be as far from complex as possible we need our model to discard redundant connections –and if possible entire building blocks– that do not contribute to the model accuracy. Thus, we have to choose a sparse prior that will gear a significant part of connections towards zero. Also, for the RT to be applicable, the distribution must be reparameterized w.r.t. an auxiliary variable, and finally the KL term must be easy to compute while also being numerically stable in order to facilitate efficient training. As recent research does point out to non-Gaussianity [11], we plan to experiment with heavy-tailed alternatives [2]

Automatic Relevance Determination

The mechanism of Automatic Relevance Determination (ARD) is a well studied subject, first introduced in the context of sparse linear regression using relevance vector machines [156, 157]. This setting causes a subset of parameters to be driven to zero. Assuming that the weights are independent and identically distributed (i.i.d.) we set the prior distributions to zero-mean Gaussian [3]:

$$p(W) = \prod_i p(w_i) \quad \text{where} \quad w_i \sim \mathcal{N}(0, \hat{\sigma}_i^2) \tag{3.6}$$

The straightforward choice for the approximate variational distribution that satisfies the conditions for applying RT is the factorized Gaussian:

$$q(W) = \prod_i q(w_i) \quad \text{where} \quad w_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{3.7}$$

The set of variational parameters to be optimized is $\phi = \{\mu, \sigma\}$. After reparameterization, we have:

$$w = \mu + \sigma\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, 1). \tag{3.8}$$

[2] We have included a short discussion, derivation and numerical results for a model with a Laplace prior in the Appendix.

[3] In the remainder of the text, we omit the upper limit on i-indexed sums and products for brevity. This will be implied equal to the total number of network weights-connections, unless stated otherwise.

The optimal hyperparameter $\hat{\sigma}$ of the prior distribution can be calculated by optimizing the ELBO:

$$\frac{\partial \tilde{\mathcal{L}}(W)}{\partial \hat{\sigma}_i^2} = 0 \Rightarrow -\frac{\partial}{\partial \hat{\sigma}_i^2} KL(q_\phi(W)||p(W)) = 0, \tag{3.9}$$

which yields the optimal parameters $\hat{\sigma}_i^2 = \mu_i^2 + \sigma_i^2$. By substituting those parameters the KL term of the ELBO can be computed analytically, as it is defined over Gaussian terms:

$$KL(q_\phi(W)||p(W)) = \frac{1}{2} \sum_i \log \left(\frac{\mu_i^2}{\sigma_i^2} + 1 \right) \tag{3.10}$$

ARD with Correlated Weights

We extended the mechanism of Automatic Relevance Determination (ARD) in order to study the correlation between the connection weights and how it affects the pruning parameters of our method. We now set the prior distributions to zero-mean multivariate Gaussian:

$$p(W) = \mathcal{N}(w|0, \hat{\Sigma}), \tag{3.11}$$

where w is now a vector containing all the connection weights of our model. The straightforward choice for the approximate variational distribution that satisfies the conditions for applying SGVB is Gaussian:

$$q(W) = \mathcal{N}(w|\mu, \Sigma), \tag{3.12}$$

and the set of variational parameters that we wish to optimize is now $\phi = \{\mu, \Sigma\}$. Reparametrizing, we have:

$$w = \mu + L\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I) \tag{3.13}$$

where $\Sigma = LL^T$ is the Cholesky decomposition of the covariance matrix Σ. The optimal hyperparameter $\hat{\Sigma}$ can be calculated directly by maximizing the ELBO which yields parameters $\hat{\Sigma} = \mu\mu^T + \Sigma$. By substituting these parameters, the KL term of the ELBO can be computed analytically. For numerical stability, the variational parameters that we estimate are the mean of the variational distribution μ and the lower triangular matrix with positive diagonal elements L^{-1}. Using the estimate matrix L^{-1}, we can sample from the variational distribution with eq. 3.13. Instead of inverting L^{-1}, we directly sample the weights by solving the triangular linear system $L^{-1}(w - \mu) = \epsilon$. Solving the linear system is much more numerically stable and can be executed fast via hardware acceleration.

In order to avoid the inversion of the covariance matrices and the calculation of the determinants during the optimization procedure we decompose the KL term as:

$$\begin{aligned} KL(q(W)||p(W)) &= -\int q(w)\log p(w)dw + \int q(w)\log q(w)dw \\ &= -\mathbb{E}_{q(w)}(\log p(w)) - \mathcal{H}(q(w)), \\ &= \frac{1}{2}\log(|\hat{\Sigma}|) + \frac{1}{2}\mathbb{E}_{q(W)}(w^T\hat{\Sigma}^{-1}w) - \frac{1}{2}\log(|\Sigma|) + C, \end{aligned} \tag{3.14}$$

where $\mathcal{H}(q(W))$ is the entropy of the variational distribution. By substituting the optimal hyperparameters found as $\hat{\Sigma} = \mu\mu^T + \Sigma$, the term $\log(|\hat{\Sigma}|)$ can be decomposed further by leveraging the matrix determinant lemma [158]:

$$\log(|\hat{\Sigma}|) = \frac{1}{2}\log(1 + \mu^T\Sigma^{-1}\mu) + \frac{1}{2}\log(|\Sigma|). \tag{3.15}$$

The second term in equation (3.14) can be computed using the reparameterization trick (3.13) which leads to the final expression for the KL term:

$$KL(q(W)||p(W)) = \frac{1}{2}\log(1 + \mu^T\Sigma^{-1}\mu) + \frac{1}{2}\mathbb{E}_{q(\epsilon)}(\tilde{w}^T\hat{\Sigma}^{-1}\tilde{w}), \tag{3.16}$$

where $\tilde{w}$ are the reparameterized sampled values for the weights using ϵ and the matrix $\hat{\Sigma}^{-1}$ can be easily computed via the Sherman-Morrison identity. This decomposition of the KL term allows us to avoid the inversion and log determinant operation in the ELBO computations leading to very stable training of the network.

3.4 Experimental Evaluation

In this Section, we provide numerical results for the proposed method, in comparison to recent existing feature fusion networks. For our numerical analysis, we perform three different experiments. First, we evaluate our methods as a backbone network for detection, using [127] and instance segmentation, using [159] versus state-of-the-art backbone combinations. We carried out experiments to evaluate how the learned architecture of our network can adapt to different types of datasets, containing objects at various scales and sizes. Furthermore, we tested the proposed probabilistic pruning methods versus different deterministic ones and we highlight the benefits of pruning components probabilistically. Finally, we introduce a way of acquiring uncertainty estimates of the model predictions and we evaluate the quality of those estimates experimentally in two different setups.

Table 3.1: Numerical results for object detection/segmentation trials on COCO [160]. Average precision and precision on different threshold and object sizes are shown, alongside with network size and inference time (measured in milliseconds), for proposed models and other feature pyramid variants.

Network	Model	AP	AP_{50}	AP_{70}	AP_S	AP_M	AP_L	Params	Inference
Faster RCNN	BiFPN	0.293	0.486	0.311	0.163	0.316	0.350	1.60M	7.8 ± 0.0
	PANet	0.296	0.486	0.314	0.167	0.320	0.351	1.74M	6.7 ± 0.0
	NAS-FPN	0.307	0.509	0.326	0.175	0.342	**0.392**	1.53M	5.4 ± 0.1
	PConv	0.308	0.510	0.320	0.180	0.346	0.391	**1.25**M	8.4 ± 0.7
	HRNet	0.305	0.510	0.310	0.161	0.345	0.381	1.32M	$\mathbf{3.2 \pm 0.1}$
	ARD	0.315	0.525	0.331	**0.187**	0.340	0.377	1.67M	6.3 ± 0.0
	FullARD	**0.322**	**0.533**	**0.342**	0.186	**0.351**	0.388	1.74M	6.5 ± 0.0
Mask RCNN	BiFPN	0.271	0.451	0.284	0.109	0.291	0.402	1.60M	7.8 ± 0.0
	PANet	0.268	0.446	0.279	0.111	0.288	0.393	1.74M	6.7 ± 0.0
	NAS-FPN	0.280	0.468	0.290	0.117	0.308	0.411	1.53M	5.4 ± 0.1
	PConv	0.279	0.464	0.290	0.117	0.309	0.410	**1.25**M	8.4 ± 0.7
	HRNet	0.288	0.484	0.301	0.114	0.314	0.418	1.32M	$\mathbf{3.2 \pm 0.1}$
	ARD	0.290	0.481	0.303	0.124	0.315	0.424	1.67M	6.5 ± 0.0
	FullARD	**0.299**	**0.499**	**0.314**	**0.126**	**0.324**	**0.447**	1.74M	6.8 ± 0.0

Table 3.2: Numerical evaluation of uncertainty estimates for Faster RCNN trained on three different datasets. *Baseline* indicates detections acquired using the weight scaling rule and thresholded via the use of NMS, *Mean* detections are obtained with test time averaging and NMS applied and *Var voting* indicates predictions of time averaging but with the use of prediction variance coupled with the var voting algorithm. Ten forward passes where performed for each image.

Dataset	Model	AP	AP_{50}	AP_{70}
PlantDoc	Baseline	0.321	0.525	0.354
	Mean	0.333	0.533	0.364
	Var voting	**0.351**	**0.539**	**0.404**
COCO	Baseline	0.313	0.521	0.332
	Mean	0.286	0.451	0.315
	Var voting	**0.341**	**0.563**	**0.365**
Cards	Baseline	0.886	0.997	0.984
	Mean	0.889	0.999	0.989
	Var voting	**0.912**	**0.999**	**0.994**

3.4.1 Implementation Details

Model Details

Following [161, 162] we use feature pyramid levels P_3 to P_7, where P_3 to P_5 are computed from the output of the corresponding ResNet-50 residual stage (C_3 through C_5) using top-down and lateral connections, P_6 is obtained via a 3×3 stride-2 convolution on C_5, and P_7 is computed by applying ReLU followed by a 3×3 stride-2 convolution on P_6. These minor modifications have a positive impact on training and inference speed while maintaining accuracy. We used a 3×3 depth-wise separable convolution [95] for feature fusion, as it reduces significantly the number of trainable parameters, and we added batch normalization and ReLU activation after each convolution. Each connection weight was restricted to positive values via the use of ReLU activation. Also, in order to avoid numerical instabilities our network optimizes the logarithmic variance $log(\sigma_i^2)$ of the variational distribution, instead of the equivalent optimization over σ_i^2. All the means were initialized with a value of 1 and the logarithmic variances were set to 0 corresponding to variance of 1. In order to improve training stability and also force our model to take advantage of all 5 levels, we added residual connections from P_3, P_4, P_5, P_6, P_7 to their respected outputs. This significantly helped the optimization procedure and slightly boosted model accuracy.

Training Details

Our main experiments are conducted on the large-scale detection benchmark COCO [160]. Following common practices [161, 162], we use the COCO trainval35k split (115K images) for training and the minival split (5K images) as validation. We report our main results on the test dev split (20K images) by uploading our detection results to the evaluation server. Our model implementations were based on the MMDetection open source project [163]. At each trial, we have trained the network for 15 epochs using Stochastic Gradient Descent (SGD) with momentum set to 0.9 and weight decay parameter set to 0.0001. The learning rate was set according to the linear scaling rule [164]; this rule states that the learning rate has to be proportional to the batch size, where each batch was set to include 2 input images, each at resolution of 1333×800 pixels. The training and test parameters for the employed detectors were set according to the values prescribed in the respective papers.

Variational Details

We prune redundant connections based on the distribution of weights w. When the means of the variational distribution are less than a threshold value, those connections are dropped; additionally, when a building block is left with zero input connections, the whole block is dropped, further reducing model complexity. Through our experiments, the KL term in the loss function was annealed by a small factor to avoid over-regularization. At test time, we followed the weight scaling rule [143] by replacing the weights with their expected values, i.e. formally:

$$\mathbb{E}_{q_\phi(w)} p(y|x,w) \approx p(y|x, \mathbb{E}_{q_\phi(w)}[w]). \tag{3.17}$$

3.4.2 Detection Predictive Performance

For fair comparison we followed the same training scheme for all the networks. As recent works have shown, repeating the same feature fusion network multiple times enables higher-level feature fusion and provides better accuracy. However, in our experiments we choose not to repeat the networks, as we believe that stacking layers makes the results of the experiments more difficult to interpret.

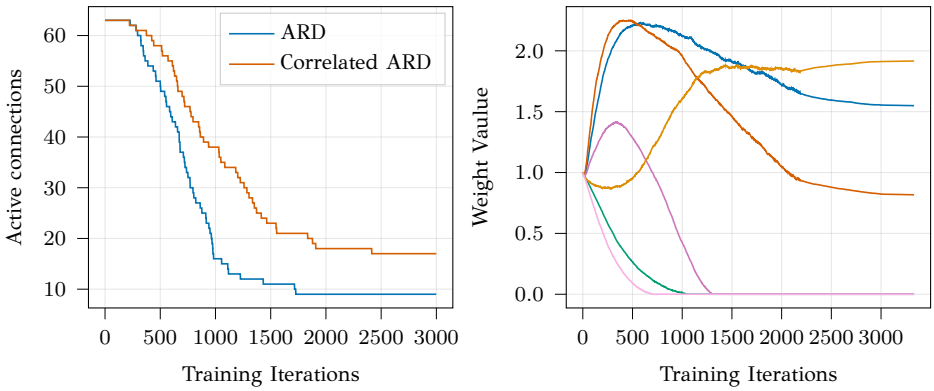

Figure 3.3: Left: Plot of the number of non-pruned weights/connections versus training iterations using different priors on the same setting (Faster RCNN on COCO). Right: Plot of indicative values of the means of the approximate posterior versus training iterations, over randomly picked network connections (each color corresponds to a different connection).

Table 3.1 shows the accuracy and model complexity for our proposed network and other state-of-the-art feature fusion networks, (NAS-FPN) [137],(PANet) [135], (BiFPN) [5], (PConv) [130], (HRNet) [165]. The variational-based networks were

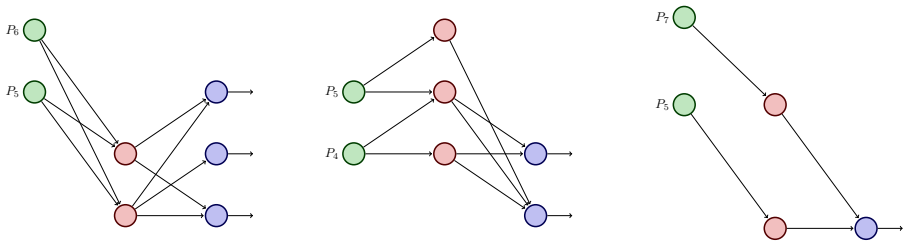

Figure 3.4: Plot of different resulting architectures for the trained model, combined with the proposed FullARD prior on Faster RCNN on three different datasets (top row: "COCO", "Plants", bottom row: "Cards").

compared according to the choice of prior distribution. As we can see, in all cases accurate detection is quite a challenging problem. It is clear that supervised segmentation techniques like deep convolutional neural networks can benefit from the presence of sophisticated feature fusion. The variational models outperform the standard architectures in terms of average precision. With respect to the type of the model prior, the model that is integrated with the correlated Gaussian (FullARD) outperforms other variants, but at a small cost of pruning less weighted connections.

Table 3.3: Numerical results for instance segmentation trials on COCO [160]. Average precision (AP) is shown, alongside with network size (in terms of preserved connections, "Cons" and number of parameters, "Params") and inference time (measured in milliseconds) for different pruning schemes.

Model	*AP*	Cons	Inference	Params
No Pruning	0.299	63	14.2 ± 0.1	1.74
Random Pruning	0.222	16	8.1 ± 0.04	1.60
Lasso-based	0.283	9	$\mathbf{4.8 \pm 0.02}$	**1.32**
Method in [166]	0.286	9	6.1 ± 0.03	1.38
Method in [167]	0.280	9	7.1 ± 0.02	1.40
ARD	0.290	**9**	6.5 ± 0.01	1.39
FullARD	**0.299**	16	6.8 ± 0.02	1.60

In Table 3.3, we added some experiments in order to highlight the benefits of pruning weights in a probabilistic manner. Specifically, we experimented with the initialized complex architecture of our model with no pruning; also, we randomly pruned a subset of weights and trained the resulting network via maximum likelihood (respectively "no pruning" and "random pruning" in Table 3.3). We tested the non-

probabilistic method of Lasso pruning where a scaled regularization term based on the L_1 norm of the weights was added to the detector loss function. Finally, we added more sophisticated deterministic pruning methods. We experimented with the gradient-based pruning method in [166] and with the method of [167]. Both methods require the fully connected network to be trained up to an optimal point, and then iterative connection pruning is applied. We set both methods to prune connections until 9 connections remain (in order to match ours and the Lasso-based pruning techniques).

We can see that the probabilistic methods of pruning yield better results than the deterministic ones. The correlated prior furthermore has the same accuracy as the fully complex model with no pruning while keeping only 25% of total connections. Additionally, both the variational and Lasso based methods yield better results than random pruning, verifying our notion that the network can learn to drop redundant connections. In Figure 3.3, we present two plots: the top plot depicts the active weights versus the training iteration, where all the methods progressively drop this number to a point where it reaches stability. In the bottom plot, we can observe some indicative values of the means of the approximate posterior on the weights.

Finally, we trained our proposed model with the ARD prior on the connection weights, integrated with the Faster RCNN network in three distinct datasets (COCO, Plants, Cards). By conducting these experiments we wanted to study the feature fusion architecture (i.e., corresponding to the non-pruned connection set after training). The datasets for this experiment where carefully chosen as each one bears its unique characteristics. Specifically, COCO [160] is an extremely demanding dataset containing multiple objects at different scales and sizes, PlantDoc [168] contains (mostly) medium and large objects and the Cards dataset [169] is comprised solely of small objects. As we can see in Figure 3.4, each dataset leads to its own distinct optimal architecture. This means that the network can learn to fuse and use those feature maps that are more valuable to each specific dataset.

3.4.3 Evaluating Predictive Uncertainty

We conducted experiments to quantify the uncertainty estimates of our model. By having the distribution of the connection weights w we can easily sample values of w. Each different sample from these distributions results to a distinct feature fusion

Figure 3.5: Results of standard NMS method (left column) and the Var voting results (right column) for the Faster RCNN network combined with the proposed method with ARD prior on COCO dataset. Bounding boxes are drawn with blue color, and the variance of each bounding box is outlined using green circles (variance is proportional to the radius around the respective bound point).

network. Thus, instead of directly using the mean values of w at test time, we can first draw sample of weights acquiring a feature fusion network and then pass the image to it. This practice is referred to as test time averaging, and has been used for acquiring uncertainty estimates in stochastic neural networks [93]. Also, it has been previously applied in an objection detection architecture context [149].

We have performed 10 single forward passes (each time sampling different values of w), a process which yields a larger set $D = \{D_1, \ldots, D_{10}\}$ of 10 individual detection sets $D_i = B, S$ where B and S are sets containing the bounding boxes and the object scores respectively. Those prediction sets will have significant overlap, thus we sort the predictions according to their IoU and then we calculate the mean and variance of each prediction box resulting in $D_f = \{B_{mean}, B_{variance}, S_{mean}, S_{variance}\}$.

In order to numerically quantify those uncertainty estimates we used the variance voting ("var voting") algorithm proposed in [7], which modifies the non-maximum suppression (NMS) scheme. It uses the variance of a predicted location and refines each candidate bounding box location according to the learned variances of neighboring bounding boxes. The results are reported in Table 3.2. For all the datasets we can observe that the use of variance estimates can slightly improve network performance.

We can observe the effectiveness of the var voting algorithm combined with our uncertainty estimates in Figure 3.5. The bounding boxes where refined to better match the target object. We can also observe model uncertainty for the predicted locations of bounding boxes, and we can see that accumulating predictions can even result in a prediction that a single forward pass could miss.

3.4.4 Segmentation Uncertainty in Low Data Regime

While FPNs are primarily used for object detection due to their ability to generate refined features, the head architecture of the proposed Variational FPN in Figure 3.2 can be adapted for various tasks. To demonstrate this flexibility, we experiment with semantic segmentation, following the approach outlined in [170]. Specifically, we explore the limits of potential benefits arising from the Bayesian interpretation of the VarFPN architecture by applying our model to the challenging task of low-data semantic segmentation.

We evaluate the model's performance on a real-world inscription localization task, where a neural network predicts image regions containing text-related information. The dataset consists of only 67 images containing inscriptions written in Greek, found within Byzantine churches and monasteries of Epirus, Northwestern Greece. This represents an extremely low-data regime by deep learning standards. Limited training data in turn, can lead to overfitting, where the model memorizes specific details of the training set and performs poorly on unseen data. Additionally, overconfidence can potentially arise, where the network assigns high certainty to inaccurate predictions on slightly different test data. This instability renders the model unreliable for real-world applications.

To avoid this behavior in practice, highly regularized networks with significant sparsity are often deployed. In this direction, Sfikas et al. [171] proposed quaternion CNNs for lightweight modeling. Quaternion operations reduce the number of parameters needed for the multidimensional representation of a single RGB pixel, leading to better image classification results and compressed models. They further proposed QGan, a quaternionic conditional GAN (inspired by[172]), to generate segmentation maps from an input image. We propose an improvement by integrating the variational FPN model with QGan. This creates a more flexible generator network that can effectively fuse extracted features from the input image, ultimately leading to

superior segmentation results. While the intrinsic sparsity of quaternions and hypercomplex networks in QGan allows for lightweight models that function well with limited training data, the probabilistic aspect introduced by VarFPN enhances the prediction segmentation maps with uncertainty estimates. Notably, the ”plug-and-play” design of our method simplifies implementation, making it a cost-effective way to ”Bayesianize” image understanding tasks.

For training this new model, we adopted the experimental protocol established in [171]. During training, we employed annealing with a small factor on the KL divergence term within the loss function to prevent over-regularization. To assess the effectiveness of the FPN Quaternionic GAN model, we compared it to its corresponding vanilla (non-FPN) counterpart. This comparison ensured both models had the same number of neurons, accounting for the additional nodes introduced by the Variational FPN.

Table 3.4: Numerical results for two variants of the proposed model (Variational FPN-QGAN) versus its counterpart with the same number of neurons (QGAN). Test BCE figures (lower is better) are shown and corresponding IoU scores in parenthesis (higher is better).

Model	Test BCE	Test IoU
Quaternion GAN	6.54	45.4%
Quaternion FPN GAN	**4.74**	**61.2%**

In Table 3.4, we compare our two model variants using binary cross-entropy (BCE) and Intersection over Union (IoU) metrics. The results demonstrate that the introduction of the VarFPN and a fusion process outperforms the standard U-Net-based model from [171]. In both metrics, the FPN generator produces higher quality segmentation results. The benefits of the probabilistic interpretation are further evident in Figure 3.6, which shows segmentation results for test images. The Bayesian approach allows us to obtain not only the mean prediction but also the variance, highlighting the model’s uncertainty in its final predictions. The importance of modeling uncertainty, particularly in areas outside of text regions, is crucial for robust text segmentation as the limited training data leads to a significant number of false positives, where the model erroneously identifies non-text regions as text. However, the model’s ability to highlight these regions with high variance indicates its own uncertainty. This variance estimate can be leveraged to effectively suppress false positives

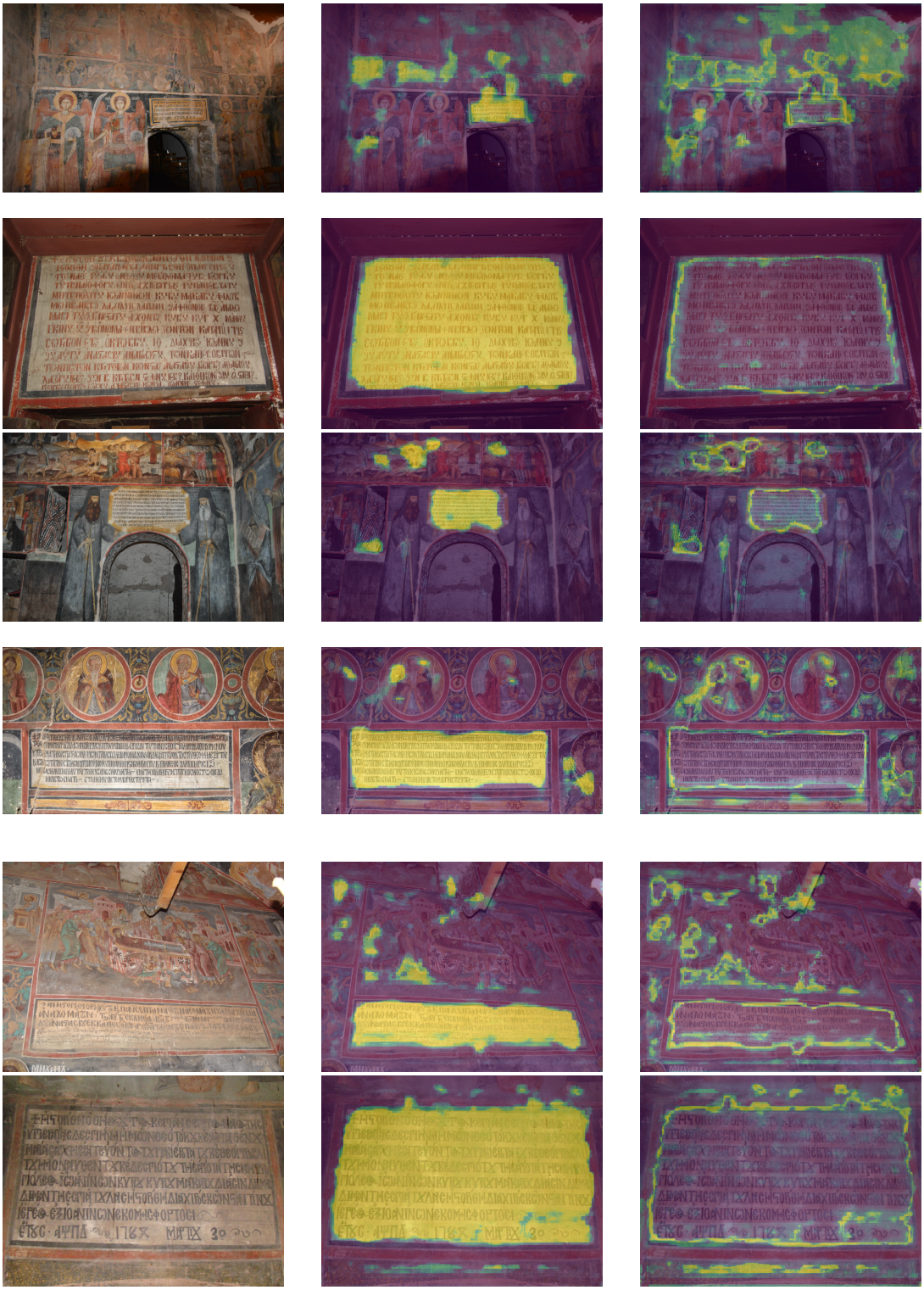

Figure 3.6: Qualitative Results for VarFPN-QGAN: The figure shows qualitative results for the VarFPN-QGAN model. The left column displays the input images. The middle column shows the mean text predictions generated by the enhanced generator network using Monte Carlo simulations. The right column visualizes the variance of these predictions.

without requiring additional training data or complex post-processing techniques.

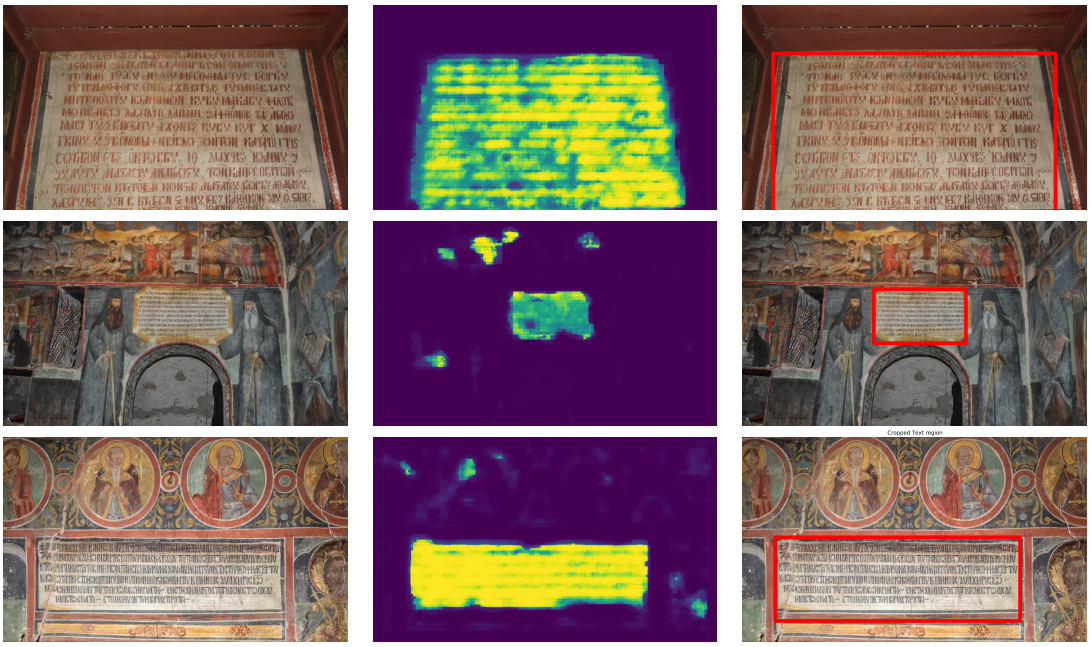

Figure 3.7: Qualitative Results: Uncertainty-Weighted Prediction for the VarFPN Model. Figure shows the effectiveness of uncertainty weighting in suppressing false positives. The left image displays the raw prediction, while the right image shows the final text region prediction after applying uncertainty weighting. The weighted prediction image uses a larger connected component to identify the true text region, effectively suppressing most false positives.

3.5 Discussion

In this chapter, we have presented Variational Feature Pyramid Networks, as extensions to the widely used FPN backbone. These networks are efficient and can be easily applied to various detectors for more efficient feature fusion. They can adapt to the underlying data, leading to specific fusion architecture for each training set. The Bayesian framework is used in our method, provides uncertainty estimates about the predictions of the trained model in contrast to deterministic variants. Numerical experiments show that the integration of the proposed model results in improving overall detection efficiency.

In the next chapter, we will turn to the problem of efficient Bayesian deep learn-

ing through low-dimensional probabilistic modeling and optimization. While the approach discussed in this chapter can provide reasonable predictive uncertainty estimates using a low-dimensional set of probabilistic parameters, selecting this set isn't always straightforward, limiting its applicability outside of detection architectures. Therefore, we will explore a method for applying Bayesian inference in a highly efficient manner, eliminating the need for manual selection of an a-priori set of parameters.

Chapter 4

Low Dimensional Bayesian Deep Learning via Implicit Neural Representation Inference

This chapter explores the challenges of applying Bayesian treatment to large deep learning models. While large models have more learnable parameters, complex architectures, and stronger inductive biases which translates to better performance, they also pose significant challenges for incorporating Bayesian techniques and probabilistic reasoning. Our approach tries to address this issue by introducing a novel method for integrating highly rich (e.g. non-linear) approximate inference into deep image models. We propose learning an efficient hypernetwork that utilizes elements from Implicit Neural Representation (INR) methods. This hypernetwork produces probabilistic variables that introduce stochasticity to the weight parameters of the main neural network. Furthermore, the overall Bayesian inference leverages the hyper-

network's parameters, which remain low-dimensional thanks to the implicit neural representation, making the entire framework computationally efficient. As Bayesian inference is performed over the low-dimensional space of hypernetwork parameters we can use richer approximations for the desired posterior distribution.

The rest of this chapter is structured as follows. We begin in Section 4.1 by discussing the challenges associated with applying Bayesian deep learning and outlining our contributions. Section 4.2 explores related works, including methods that utilize low-dimensional Bayesian deep learning, those that leverage hypernetworks for probabilistic design similar to our approach, and stochastic implicit neural representation methods. In Section 4.3, we discuss our proposed INR hypernetwork modeling and the integration of highly flexible posterior approximation methods into our scheme. Finally, Section 4.4 presents the experimental evaluation of our method.

4.1 Challenges of Bayesian Deep Learning & Contributions

Bayesian Neural Networks (BNNs) are a class of models that propose elegant solutions to the pathologies of standard NNs [173, 174, 175]. In BNNs, model parameters are defined as random variables that follow a prior (posterior) distribution, which encodes knowledge about the model before (after) having "seen" the training data. Learning is cast as an inference problem, where the task is to compute efficiently the posterior distribution. In turn, making predictions on new data is replaced by computing a predictive distribution. Advantages include that uncertainty estimates are calibrated and robust, and hyperparameter estimation can be performed through a principled evidence maximization framework. In BNNs, Bayesian inference is not exact, and a direct application of Bayes' law leads to an intractable computation. An approximation has to be applied, and in this respect numerous solutions have been proposed. A factor that complicates this problem is that the approximation must lead to a scalable, practical implementation that must take into account that the data and model size may be far larger than what was the norm in methods and models that dominated Bayesian inference in the pre-deep learning era. Several solutions have been proposed in this respect, rehashing and adapting older methods [8, 9] or putting forward completely fresh approaches [10].

Scalability is a crucial factor when it comes to learning methods in the context

of NNs. Assuming an entire network to be probabilistic implies significant overhead in terms of various factors. Common remedies include assuming a Gaussian form combined with a low-rank approximation of the Hessian, and using a simplified, even diagonal covariance structure. Kronecker-Factored Approximate Curvature (KFAC) expresses a useful tradeoff, which neglects only cross-layer correlations and uses a block-diagonal covariance matrix [43]. Another option involves treating only part of the network as non-deterministic [34, 120]. We then have uncertainty only in the last layer neurons, treating the rest of the network as a feature extractor [119]. As a consequence, and to the degree that these assumptions are overly simplistic, the approximate distributions may turn out to be very far from the actual posterior and predictive. This often translates to a dramatic reduction of predictive strength in practice.

Implicit Neural Representations are related to a different line of research that is orthogonal to that involving Bayesian networks [2, 176]. With INRs, the goal is to represent a signal in terms of a trained neural network. Unlike standard representations as discrete sets of values over a canonical grid, an INR accepts continuous coordinates as inputs. Therefore, the INRs allow for a continuous representation, with the underlying NN providing values of the represented signal at theoretically any granularity. Related breakthroughs in improving representation of high frequencies have contributed to the popularity of the approach [2, 177]. Numerous signal representation use-cases have been explored, including images, video, 3D shapes or Neural Radiance Fields (NeRFs). With the latter, a NN is tasked with mapping ray position and direction to color and density values. Part of the parameters of a larger NN can also be encoded with an INR; in [178], convolutional kernels are represented in terms of Multiplicative Anisotropic Gabor Networks. In this case, the implicit representation allows for kernels that generalize well to higher resolutions than the ones originally trained with. Aside from allowing for continuous representation at multiple scales, another major focus involves the INR's capability of producing a compressed, low-dimensional representation [179, 180].

In this work, we propose a class of Bayesian Neural Network that is parameterized using a combination of deterministic and stochastic parameters. In recent work, similar partitions are employed [34, 123, 120], where a specific subnetwork is set to be stochastic while the rest of the network is deterministic. Unlike these works, we define *all* parameters as functions conditioned over both deterministic and probabilis-

tic components. Normally, this is very much desired but computationally prohibitive due to the huge number of parameters in modern NNs; in our work, this is made feasible due to the probabilistic component being parameterized through an INR hypernetwork, which compresses probabilistic factors through a low-dimensional SIREN representation [2]. It is over this representation that we assume a prior distribution, and perform inference. As the number of probabilistic factors is kept low, we are allowed to make fewer concessions w.r.t. constraining the form of the posterior and the predictive. The result is a process that is comparatively closer to exact inference, leading to more accurate estimates and better uncertainty calibration.

In a nutshell, the deterministic model component is responsible for ensuring accurate results, while the low-dimensional probabilistic component is responsible for inducing stochasticity to the entirety of the network. We validate our claims and model across a variety of experimental trials, where we show that our model produces accurate and well-calibrated uncertainty estimates.

4.2 Related Work

This section examines related works exploring methods to introduce flexible approximate posterior distributions into deep neural networks. We focus on approaches that introduce stochasticity to large neural networks by employing a hypernetwork design. We also consider works that explore stochastic INRs in a diverse context.

4.2.1 Low-Dimensional Inference

Bayesian inference in a low-dimensional space is another important related concept, with often considerable overlap to works that can be understood as forms of hypernetworks. [123], in the spirit of [124], employ rank-1 multiplicative noise components, before attempting to estimate an approximate posterior over the weights. [113] adopt *post-hoc* Bayesian inference by constructing a subspace of the BNN weights. They apply high fidelity inference on these small subspaces, and were able to produce state-of-the-art results at a moderately low computational cost. [122] learn a non-linear latent representation of network weights. Another subgroup of related work can be described as selecting a portion of the BNN parameters to be treated as random variables, and leaving the rest of the model to work deterministically. One of

the most popular and straightforward approaches are last-layer BNNs. By selecting *a priori* only the last layer to have a probabilistic treatment, they resort to a linear model which ensures analytical tractability of both inference and predictive distribution in the spirit of Gaussian processes, while the remaining NN structure acts as a feature extractor [116, 117, 118, 119]. Finally, [120] first obtain a MAP estimate of all weights, then define a subnetwork selected in a way that aims to maximally preserve predictive uncertainty. The small size of the subnetwork allows for the use of a full-covariance Gaussian posterior in tandem with linearized LA [181].

4.2.2 Hypernetwork Modeling

Hypernetworks are NNs that are used to predict deterministically the parameters of another, typically larger network, termed the "primary" network. The terminology is due to [105], however the main idea can be traced back to earlier works (see discussion in e.g. [106, 182]). [106] have been among the first to extend hypernetworks to a Bayesian setting. Their Bayesian hypernetwork, modelled as a normalizing flow, learns to predict distributions of weights for the primary network. The flow predicts scaling per-neuron factors for the primary network weights. This is similar to the closely related [112], which however require an extra inference network to estimate the entropy term of the VLB. Almost concurrently, [83] proposed BbH for VI with implicit distributions. They use a discriminator network for density ratio estimation (DRE) in the context of prior-constrastive VI [183], and a generator to model the variational distribution. [184] use a kernel method for DRE instead of a discriminator. [182] and [185] explore hierarchical prior modeling using NN-based implicit distributions and Gaussian processes. INRs have also been used for approximating model parameters of deep NNs [178, 186].

4.2.3 Stochastic Implicit Neural Representations

INRs have been used as models for signal compression [187], and more recently they have been extended to the Variational Bayesian setting [188]. [189] extend NeRFs to learning distributions of all possible radiance fields. A simple variational posterior is assumed, and the base model is extended to learn uncertainty estimates over scene parameters. [190] use a BNN as an INR of computerized tomography.

4.3 Implicit Neural Representation Inference

This section studies in great detail our method for applying low-dimensional Bayesian inference within deep neural network models. We propose to move from the high-dimensional setting of full inference in a modern Neural Network to low-dimensional inference, by assuming an auxiliary Implicit Neural Representation alongside the main network. We perform density estimation over the parameters of the INR hypernetwork, while treating the factors corresponding to the original weights as deterministic parameters. This allows us to employ powerful inference methods (we discuss LA, SWAG, NFs) with minimal approximation concessions, by leveraging on the small size and representational strength of the INR.

4.3.1 Implicit Neural Representation Modeling

Given the NN g_w is a mapping $g_w : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$, the first step of our approach is to augment each weight w with a multiplicative nuisance factor ξ [143, 93, 112]. In particular, we use $w \circ \xi$, where $\circ$ is point-wise multiplication, and the dimensionality of ξ is identical to that of w.

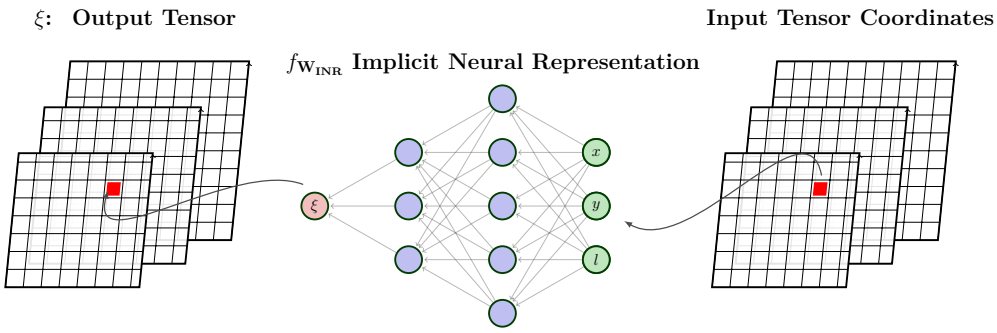

Figure 4.1: The figure illustrates the INR Hypernetwork. It takes three input coordinates, denoted as I_x, I_y, and I_l, as input. I_x and I_y represent the spatial dimensions (x and y coordinates) of the value to be generated. I_l indicates the specific layer matrix to which the generated value ξ belongs.

The ξ factor is parameterized using an INR [191], obtained as the output of a function $f_{w_{INR}} : I \to \mathbb{R}$, where tensor coordinates (domain I) are mapped to layer values. More specifically for a convolutional main network, the INR hypernetwork learns a mapping from a 5 dimensional I to a scalar value which corresponds to

the nuisance factor associated with the weight $w_{c,o,k_i,k_j,l}$ located at the kernel position k_i, k_j at channel c of filter o in layer l of the main/primary network (in the case of a fully-connected layer Figure 4.1, dimensions k_i and k_j are omitted). With the above modeling choice, the hypernetwork can be easily shared across each layer of the main network and reduce the overall modeling complexity. The architecture of the INR is defined as a multi-layer perceptron with sinusoidal activations, as with the SIREN model of [2]. Formally, the input vector z_{i-1} for layer i is transformed according to $z_{i-1} \rightarrow sin(\omega_0(w^i_{INR} z_{i-1} + b^i_{INR}))$, where w^i_{INR}, b^i_{INR} denote weights and biases of the INR layer i, and ω_0 is a fixed hyperparameter.

In INRs, any target quantity can be modelled regardless of its size, while in traditional networks parameter size is coupled with target dimensionality. This characteristic, in combination with the stochastic character of ξ allows us to choose the complexity of $f_{w_{INR}}(\cdot)$ to be (much) lower than that of its target ($d_{w_{INR}} \ll d_\xi$). Thus, w_{INR} parameters can also be interpreted as a low-dimensional representation of factors ξ.

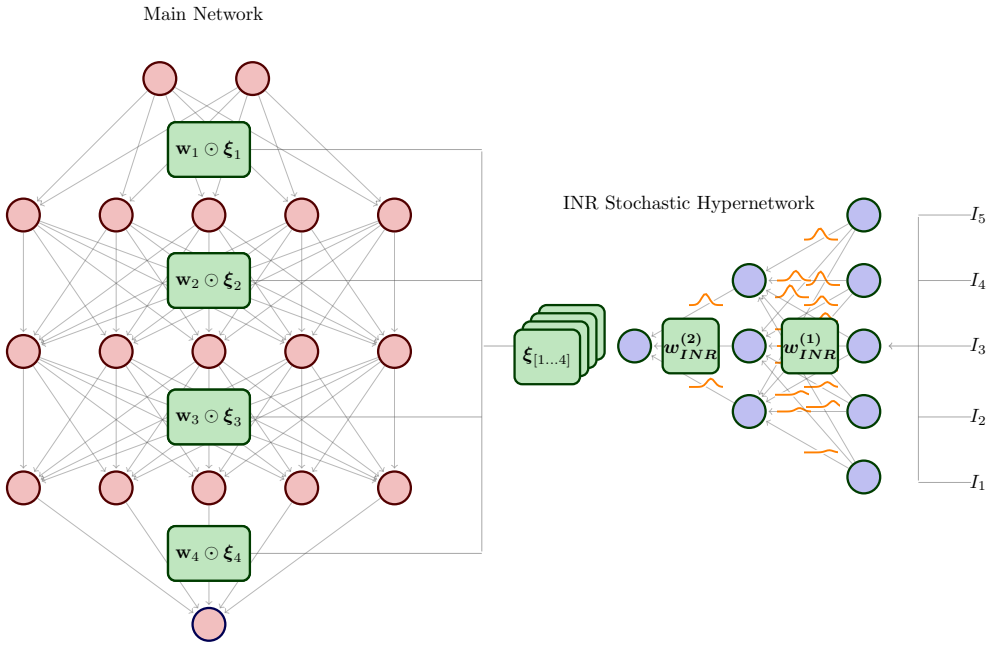

Figure 4.2: Illustration of the proposed INR model.

4.3.2 Bayesian Inference over the Neural Representation

In our method, we treat the product $w \circ \xi$ as a stochastic random variable coming from a parametric distribution $p(w, \xi) = p(w)p(\xi)$. Here we are taking advantage of the INR hypernetwork modeling of ξ and implicitly place a prior over those variables, by defining a prior over the INR parameters w_{INR}. This allows us to reason about ξ but in the much lower dimensional space of w_{INR}. Following the supervised learning setting of Section 2.2.1, our aim remains to compute the posterior $p(w, w_{INR}|D)$. Since the posterior distribution cannot be obtained in closed form, we cannot apply exact inference methods. Thus we resort to approximate inference, under an additional assumption that we only require a deterministic estimate over w. We encode this constraint as a factorization over separate approximate posterior distributions $q(w)$ and $q(w_{INR})$, where $q(w) = \delta(w - \underline{\mathbf{w}})$, and $\delta(\cdot)$ is the Dirac delta function. This forces w to be deterministic, equal to a point estimate $\underline{\mathbf{w}}$. The full approximate posterior is then written as:

$$p(w, w_{INR}|D) \approx q(w, w_{INR}) = q(w_{INR})q(w) = q(w_{INR})\delta(w - \underline{\mathbf{w}}). \tag{4.1}$$

Laplace Approximation

One way to proceed is by constructing a Laplace approximation over $q(w_{INR})$. We approximate $p(w_{INR}|D)$ by

$$q(w_{INR}) = \mathcal{N}(\bar{w}_{INR}, \Lambda^{-1}), \tag{4.2}$$

$$\Lambda = C^{-1} + \sum_{n=1}^{N} \nabla^2_{w_{INR}} \log p(y_n|g_{w,w_{INR}}(x_n))|_{\bar{w}_{INR}}, \tag{4.3}$$

where we have assumed a prior $w_{INR} \sim \mathcal{N}(0, C)$. Mean $\bar{w}_{INR}$ is found as the Maximum a Posteriori solution (Section 2.2.3). Under this scheme, $q(w, w_{INR})$ is expressed by a product of a Gaussian and a Dirac delta distribution, which can be seen alternatively as a single Gaussian distribution with precision $\gamma \to +\infty$ for variates corresponding to w and zero covariance between w and w_{INR} terms by assumption (eq. 4.1). Concerning the weights and biases that directly parameterize the "main" network (i.e. the product $w \circ \xi$), we note that these are in general non-Gaussian, even under LA assumptions. The INR $f_{w_{INR}}(\cdot)$ transforms the (approximately) Gaussian w_{INR} into a non-Gaussian density $q(\xi)$. This is multiplied by deterministic w where

the result follows a density that is a scaled version of $q(\xi)$. The first and second moments are equal to $W\mathcal{E}\{\xi\}$ and $W\mathcal{V}\{\xi\}W$, where $W = diag\{w\}$ and $\mathcal{E}\{\cdot\}, \mathcal{V}\{\cdot\}$ denote expectation and covariance respectively. Once we have computed a posterior over the weights, we can estimate the predictive (eq. 2.2) by acquiring ξ samples by first sampling $w_{INR} \sim q(w_{INR})$ and evaluating $\xi = f_{w_{INR}}(\cdot)$. We finally scale them by w, then the product is used to compute $g(x)$ and $p(y|g(x))$ in a Monte Carlo fashion.

Alternatively, the predictive distribution (eq. 2.2) can be computed in closed form, as long as we impose a linearizing assumption over the network output. Specifically, this involves a first-order Taylor expansion of network output $g(\cdot)$ around w_{INR}. As by LA assumption, parameters w_{INR} are *a posteriori* Gaussian-distributed, a linear transformation over them through linearization would result in a Gaussian predictive as well; linearization over other variables (w, ξ) would not have been fruitful due to their being non-Gaussian. Hence, we only require parameters w_{INR} to vary in this approximation, while we assume the rest of the parameters w to be constant at their MAP solution. Formally we write:

$$g^{lin}(x) \approx g_{\bar{w},\bar{w}_{INR}}(x) + J_{w_{INR}}(x)(w_{INR} - \bar{w}_{INR}), \tag{4.4}$$

where we used $J_{w_{INR}}(x) = \frac{\partial g_{\bar{w},w_{INR}}(x)}{\partial w_{INR}}|\bar{w}_{INR}$. For the predictive we then have:

$$p(y|D, x^\star) = \mathcal{N}(g_{\bar{w},\bar{w}_{INR}}(x^\star), J^T_{w_{INR}}(x^\star)\Lambda^{-1}J_{w_{INR}}(x^\star)). \tag{4.5}$$

Stochastic Weight Averaging

An alternative over LA is to use SWAG [10] over INR parameters. In this context, this amounts to approximating $p(w_{INR}|D)$ by a Gaussian $q(w_{INR})$ as in eq. 4.2, but with inverse Λ equal to the sample covariance over the SGD trajectory:

$$\Lambda^{-1} = \frac{1}{T-1}\sum_{i=1}^{T}(w^{(i)}_{INR} - \bar{w}_{INR})(w^{(i)}_{INR} - \bar{w}_{INR})^T, \tag{4.6}$$

where $\{w^{(1)}_{INR}, w^{(2)}_{INR}, \ldots, w^{(T)}_{INR}\}$ are training updates of INR parameters. The predictive distribution is estimated by Bayesian model averaging through Monte Carlo sampling. Formally we have:

$$p(y|D, x^\star) \approx \frac{1}{K}\sum_{k=1}^{K} p(y|g_{\bar{w},\xi_k}(x^\star)), \tag{4.7}$$

where K samples $\{\xi_1, \xi_2, \ldots, \xi_K\}$ are drawn from the approximate posterior $q(\xi)$ by evaluating $w_{INR} \sim q(w_{INR})$ as described in the previous paragraph.

Normalizing Flows

Normalizing Flows are another powerful modeling choice for $q(w_{INR})$. In this context, $q(w_{INR})$ is freed from the Gaussian restriction and can be any parameterized flexible parametric distribution. A normalizing flow transforms an initial random variable z, typically sampled from a standard Normal, by applying a chain of invertible parameterized transformations. The RealNVP model [192] is based on a flow composed of a series of affine coupling layers defined as: $y \rightarrow m \circ z_{i-1} + (1-m) \circ (z_{i-1} \exp(s(m \circ z_{i-1})) + t(m \circ z_{i-1}))$ where s and t stand for scale and translation, which are typical linear mappings, while m is a channel-wise masking scheme. The flow parameters can be computed by directly optimizing the variational lower bound:

$$L(w, w_{INR}) = \mathbb{E}_{q(w_{INR})} \log p(y|g_{w,w_{INR}}(x^{\star})) - KL(q(w_{INR})||p(w_{INR})), \tag{4.8}$$

where the carefully designed coupling layers ensure that the inverse and the Jacobian of the determinant of each transformation can be efficiently computed. The predictive distribution is estimated by Bayesian model averaging through Monte Carlo sampling similar to eq.4.7.

Table 4.1: Numerical results for classification on CIFAR10 (top) and Corrupted CIFAR10 (bottom) for different design choices. Log-Likelihood (↑) and Expected Calibration Error (↓) are reported.

	Modeling			Noise Structure			Type of INR		Noise Type		Activation Type	
	w	wξ	ξ	Rank-1	Channel	Full	Individual	Shared	Mult	Add	ReLU	Sine
LL	-1.29	**-0.37**	-0.44	-0.47	-0.40	**-0.37**	**-0.29**	-0.37	**-0.37**	-5.28	-0.289	**-0.287**
ECE	**0.01**	0.05	0.06	0.06	**0.05**	**0.05**	0.06	**0.05**	**0.05**	0.19	**0.032**	0.034
LL	-1.80	**-0.97**	-1.60	-1.43	-1.18	**-0.97**	**-0.90**	-0.97	**-0.97**	-6.29	-0.50	**-0.47**
ECE	0.17	**0.11**	0.20	0.20	0.15	**0.11**	**0.10**	**0.11**	**0.11**	0.26	**0.06**	0.05

4.4 Experimental Evaluation

In this Section, we provide numerical results for the proposed INR-based scheme, in comparison to recent Bayesian inference methods. Namely, we compare ourselves versus methods that perform inference over the full network, (i.e. the full set of network parameters): MC Dropout [147]; Bayes by Hypernet (BbH) [83] Deep Ensembles [54]

– considered among the state-of-the-art methods for uncertainty estimation in Deep Learning [193, 98] – and last layer Laplace approximation (LL). Albeit we consider additional baseline methods that apply low-dimensional inference in order to isolate the INR "subspace" contribution in the predictive performance of our method.

We start by experimenting with different types of modeling choices and evaluate each one on a baseline classification task, in order to quantify how our method performs under different modeling scenarios. For our main numerical analysis, we deployed three different experimental setups. First, we evaluate our predictive uncertainties for our method on a $1D$ synthetic regression task. We carried out experiments to evaluate INR performance on different types of regression UCI datasets. Last, we ran image classification trials (CIFAR100, CIFAR10 and MNIST) where we compare ResNet variants for prediction and out-of-distribution robustness. We test three variants of the proposed INR-based model, namely INR-Laplace (eq. 4.2,4.3,4.5), INR-SWAG (eq. 4.2,4.6,4.7) and INR-RealNVP (eq. 4.8). The three variants differ w.r.t. the approximation strategy for the posterior and the predictive. For the first two cases we compute the *full Gaussian covariance for the weight posterior* (avoiding e.g. KFAC or low-rank approximations [9]). Throughout our experiments, we found that the proposed model provides good predictive uncertainties on a variety of settings, highlighting the benefits of low-dimensional Bayesian inference. Concerning implementation details of the proposed and compared models and benchmark setup in general, we have moved additional information to the Appendix (App. B.2).

4.4.1 Hypernetwork Design Choices

In this Section we carry out ablation studies that justify the particular modeling and INR architecture described in subsection 4.3.1 and help us understand the behavior of the hypernetwork under different settings. We numerically evaluate each different potential modeling scenario by training a ResNet-20 model at CIFAR-10 according to subsection:4.4.4 and evaluate its MAP solution in both in and out of distribution data. Table 4.1 includes the main results.

Our first ablation study aims to justify the introduction and use of ξ variables i.e. we investigate how the BNNs perform with only the INR for the posterior (see Table 4.1 under the column "Modeling"). As the ξ variables only serve to induce stochasticity, removing weights w result in a model which is not able to capture any

information from the training data. Furthermore, augmenting w with ξ results in a more sophisticated model which yields better calibrated predictions. We choose the INR hypernetwork to be shared across all the layers of the main network. Sharing the INR hypernetwork, besides being efficient, can also reduce significantly the dimensionality of w_{INR}, as the total $d(w_{INR})$ for the individual hypernetwork will by a multiple of the number of layers of the main network. As an example, for Wide-ResNets the magnitude of this figure can be up to hundreds of variates. Despite having less parameters, the shared version of the hypernetwork is highly comparable to its more expensive counterpart as Table 4.1 (column labelled as "Type of INR") indicates. Also, we introduce independent nuisance factors ξ for every single weight w. In Table 4.1 (column labelled as "Noise Structure") we measure the benefits of our full-rank multiplicative noise versus other low-rank modeling options used in related works [123, 112]. In the same Table (column labelled as "Noise Type"), we can see results for evaluation of two different types of noise injection in the main model, namely multiplicative noise ("Mult") and additive noise ("Add"). The additive noise hugely underperforms where multiplicative noise factors seem to provide good and calibrated solutions. Because in the multiplicative structure during training $\nabla\xi$ depends on W, we argue that as W is responsible for fitting the data, it can pass valuable information to the hypernetwork weights in the multiplicative case leading to significant increase in overall performance. Furthermore we find that Sine/Periodic activations – the "default" choice in [2] – slightly outperforms a hypernet with ReLU activations as we can see in Table 4.1 (column labelled as "Activation Type"), even though results are still very close.

Finally, we evaluate the effects of INR network size on uncertainty estimates. We want to measure how increasing the number of paramters of the hypernetwork will affect the predictive behavior of the model. We trained 3 different INR models, with increasing numbers of trainable parameters.

Following [194] and [123], in Figure 4.3 we examine the normalized diversity of INRs of increasing size, where the posterior over $w \circ \xi$ was estimated via INR-SWAG and INR-MAP. Increasing the size of the INR hypernetwork results in more complex weight posteriors, which is depicted with better scores across all metrics in out-of-distribution data. Nevertheless, a small INR with only 350 trainable parameters is competitive in this training setup.

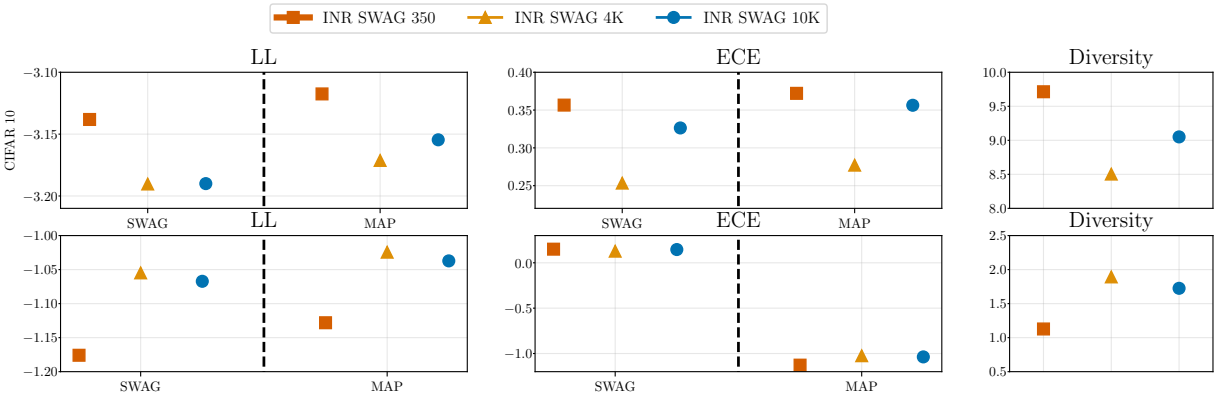

Figure 4.3: Comparison of Log-Likelihood (↑), Expected Calibration Error (↓) and Normalized Diversities between INR networks of increasing size, over CIFAR10 and corrupted CIFAR10 datasets. INR-x represents an INR with x parameters.

4.4.2 Visualizing Predictive Uncertainty

We use a synthetic $1D$ regression task with three disjoint clusters of input data as proposed in [113]. This dataset is carefully designed to test "in-between" uncertainty, i.e. model confidence in between these disjoint clusters of data [195]. This simplistic experiment is commonly used for testing model uncertainty calibration [141]. Ideally, we want a model to predict high uncertainty values as test data move away from the observed data. In this test, we use a fully-connected architecture with hidden layers

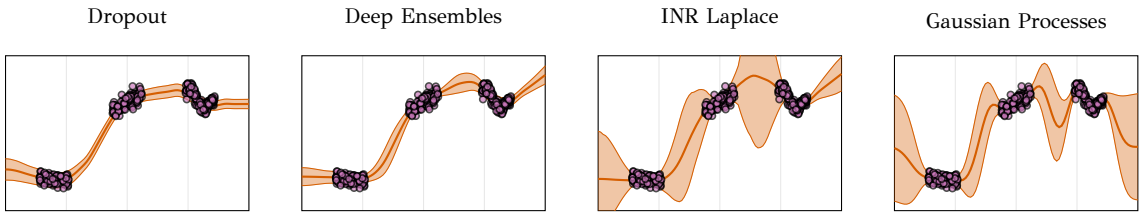

Figure 4.4: Visualization of the predictive distribution for the "toy" regression task. The data are denoted as purple circles, predictive mean is the solid orange line and the shaded region is $\pm$ 1 std.

that have $[200, 50, 50, 50]$ neurons respectively. Following [113], the network takes two inputs $\hat{x} = (x, x^2)$ and outputs a single real value $y = f(\hat{x})$. The INR network has 3 layers consisting of $[2, 10, 4]$ neurons respectively, resulting totally in 160 training parameters (equal to only 1% of the number of the ξ parameters). Results are shown in Figure 4.4. We also include a Gaussian Process with a Radial Basis Function (RBF) kernel as the state-of-the-art for this problem. Our INR-Laplace preserves

more of the uncertainty regarding both "out" and "in-between" of the observed data. Other methods, like Deep Ensembles and MC Dropout infer a desirable uncertainty structure but still remain quite overconfident. Furthermore, the proposed INR model is able to maintain the appealing characteristics of the approximate inference methods applied, specifically the stationary structure (or in-between-uncertainty) benefits of the Linearized Laplace approximation as shown in multiple works [34, 120].

4.4.3 Calibration Evaluation on Regression Benchmarks

We next test our method on the UCI regression tasks [196]. We experiment with 8 UCI regression datasets using standard training-evaluation-test splits from [63] and their GAP-variants [195]. To measure performance we deployed Gaussian test log-likelihood (LL). Our training strategy follows the work of [120]. The INR network has 4 layers consisting of $[5, 5, 5, 1]$ neurons respectively, resulting totally in 70 training parameters (equal to only 2% of the number of the ξ parameters)

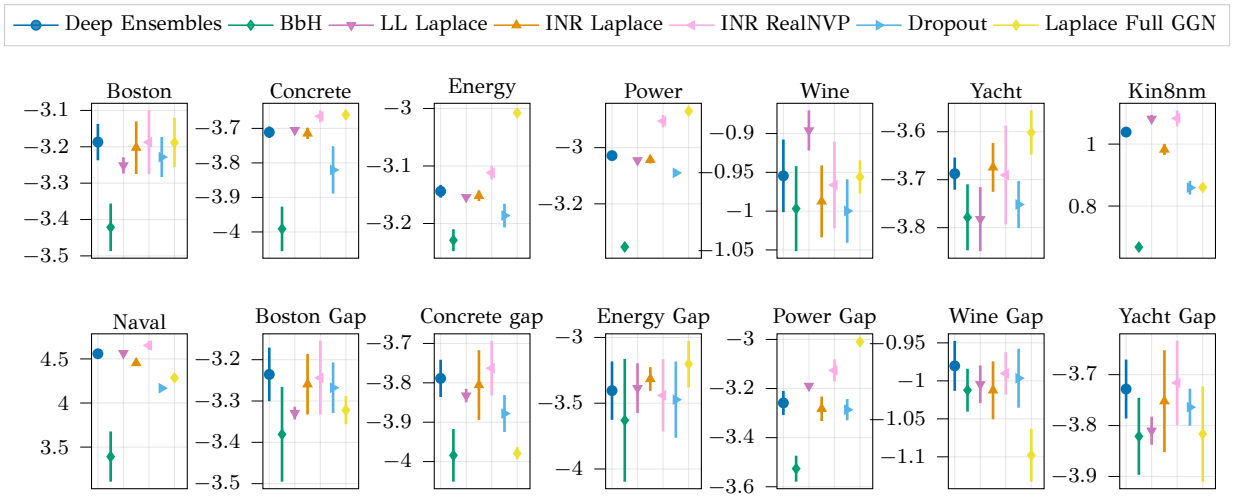

Figure 4.5: Numerical results for regression trials on UCI datasets [196]. Mean values of test Log-Likelihood (↑) are shown with ± 1 standard deviation error bars, obtained over standard [63] and GAP [195] splits.

The main results are depicted in Figure 4.5. The small MLP network enabled us to compute the full GGN matrix in the Laplace approximation of the main network and add it as baseline. As we can see, INR combined with RealNVP or LA achieves better test log-likelihood – a metric which considers both uncertainty and accuracy – compared to BbH and LL Laplace approximation, while followed closely by MC

Dropout. Furthermore, the proposed INR remain competitive with Deep Ensembles networks, even surpassing them in five out of eight datasets while overall being close enough, in both standard and gap splits, as standard deviation bars indicate.

4.4.4 Image Classification under Distribution Shift

We evaluate our method on standard image classification tasks over the CIFAR10, CIFAR100 [197] datasets. We use ResNet-50 [20] in order to test the ability of the proposed INR-based method to scale into larger models. A capable Bayesian inference technique is critical when applied in deep models, as they tend to exhibit less accurate calibration in this context [32]. We provide experiments in a context of high degree of distribution shift, as under these conditions the evaluation of predictive uncertainty is the most useful in practice [193]. The majority of methods yield good results when input data are "close" to the training data distribution, but can fail under even a mild shift in the input data [193]. Our INR hypernetwork [2], has 4 layers with $[10, 10, 10, 1]$ neurons each, resulting in 260 training parameters (only 0.001% of the parameters ξ). Following [193, 141], we train ResNet50 on CIFAR10/CIFAR100 and evaluate on

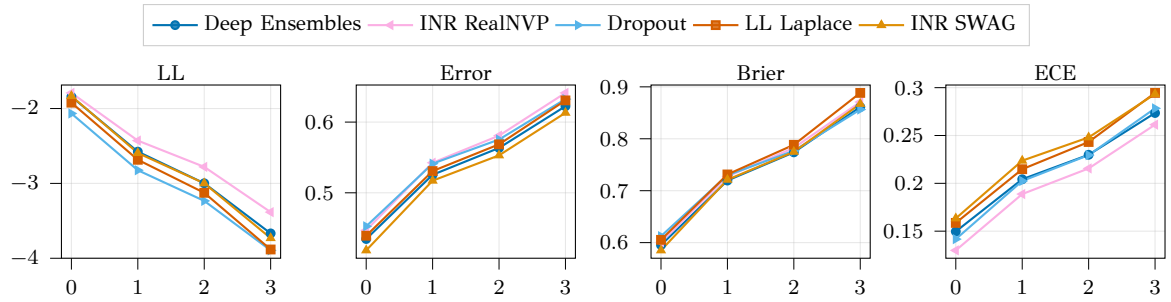

Figure 4.6: Numerical results for classification trials on Corrupted CIFAR100 dataset. The x-axis of each plot corresponds to increasingly corruption levels.

data subject to 16 different types of corruption with 5 levels of noise intensity each [198]. As Figure 4.6 indicated, one of the proposed variants, INR-RealNVP, outperforms non-INR methods in terms, log-likelihood and expected calibration error. Both INR-based methods outperform LL Laplace and MC Dropout which are overconfident in their predictions and more often erroneous while still being competitive w.r.t Deep Ensembles. Overall, these results suggest that the proposed approach produces more calibrated and accurate models than other popular uncertainty quantification approaches.

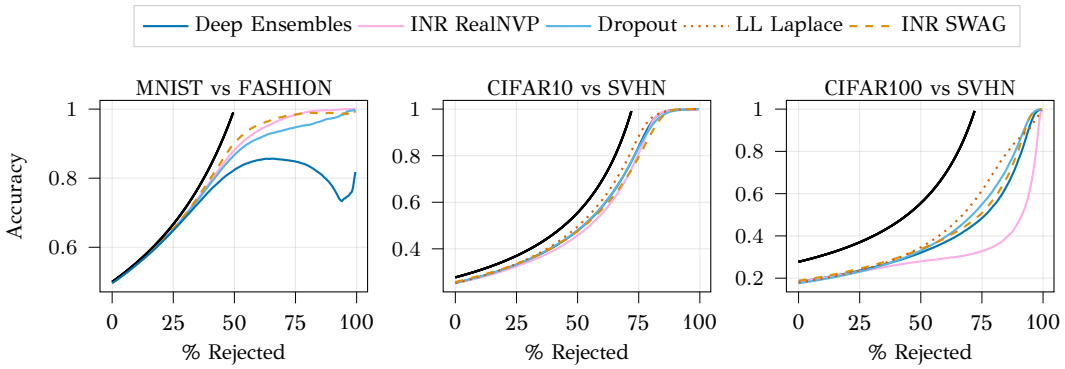

Figure 4.7: Testing the quality of calibration with Rejection-Classification plots. MNIST & CIFAR10 are considered as "in-distribution", Fashion-MNIST & SVHN are "out-of-distribution" respectively. Methods reject increasing data proportions based on predictive entropy before classifying the rest. All predictions on OOD samples are treated as incorrect. The black curve denotes maximum theoretical performance.

We quantify the quality of uncertainty estimates by jointly evaluating the predictive entropy of our model on an in-distribution and an OOD test set. Ideally, we want predictive entropy to be high on OOD data as predictions should be more uncertain, and vice versa. Following [141, 199], we deployed the OOD rejection scenario by jointly evaluating the entropy of our model on an in-distribution and OOD test set, where we allow the models to reject an increasing proportion of the data based solely on their entropy values. Ideally, we want highly calibrated and robust models to reject all the OOD examples, as well as the in-distributional examples when the corresponding predictions are inaccurate. Figure 4.7 illustrates on what percentage of the remaining non-rejected examples the predictions are accurate. On CIFAR10-SVHN all methods have the same performance, while on CIFAR100 the INR-RealNVP model fails to distinguish very uncertain in-distribution data from low uncertainty OOD ones. On MNIST-Fashion, the proposed methods INR-SWAG and INR-RealNVP perform best, followed by LL Laplace and Dropout.

Finally, we tried to measure the quality of proposed low-dimensional spaces in terms of predictive uncertainty. Specifically, we compare our INR low dimensional space with: rank-1 [123], Wasserstein subnetwork [120], and partially stochastic Resnets from [121]. We trained (each method) combined with a Resnet18 for 100 epochs in CIFAR100 while keeping the approximate inference method the same across all low-dimensional spaces. Results in Table 4.2 show a trend in favor of both

Table 4.2: Numerical results for classification trials on CIFAR100 for different proposed low-dimensional spaces alongside their inference time.

Subspace	Inference	Standard				Corrupted				Time ↓
		LL ↑	Error ↓	Brier ↓	ECE ↓	LL ↑	Error ↓	Brier ↓	ECE ↓	
Rank1	SWAG	−2.29	0.34	0.55	0.22	−4.77	0.57	0.92	0.39	0.28
	Laplace	−4.01	0.31	0.97	0.66	−4.25	0.58	**0.97**	0.40	0.55
INR	SWAG	**−2.09**	**0.30**	0.50	0.22	−4.18	0.53	0.84	0.36	**0.11**
	Laplace	**−3.91**	**0.30**	**0.96**	0.67	−4.19	0.58	**0.97**	**0.39**	0.51
Subnetwork	SWAG	−2.14	**0.30**	**0.49**	**0.20**	**−3.97**	**0.51**	**0.82**	**0.34**	0.29
	Laplace	−3.95	0.32	**0.96**	0.65	**−4.13**	**0.51**	**0.97**	0.46	**0.42**
Partially Stochastic	SWAG	−2.14	**0.30**	**0.49**	**0.20**	**−3.97**	**0.51**	**0.82**	**0.34**	0.28
	Laplace	−3.99	0.34	0.97	**0.63**	−4.18	**0.51**	0.98	0.47	0.49

proposed INR-x methods and validate to a considerable degree the premise of our method: instead of choosing a subset or subnet following the rationale of the corresponding methods, the INR produces ξ outputs that endow the full network with the desirable stochasticity, while keeping the dimensionality of the random process that we want to do inference upon at a low level.

4.5 Discussion

In this chapter, we have presented an approach for scalable and efficient Bayesian Deep Learning, that leverages on the small size and representational strength of INRs. As Bayesian inference is performed over the low-dimensional space of INR parameters we can use richer approximations for the desired posterior distribution. *In large models it is better to introduce multimodal stochasticity implicitly to the weights rather that trying to directly model them probabilistically* This method is efficient and can be easily applied to various networks for more efficient and richer Bayesian inference. Our claims are corroborated by the reported experimental results, which show that the integration of the proposed method results in improving considerably overall uncertainty estimates.

While full rich posterior approximation methods offer significant potential, as discussed in this chapter, approximate inference often leads to simpler solutions. However, this simplicity can sometimes come at the expense of not fully exploiting the richness of the approximate posterior method. In the next chapter, we study

this phenomenon by examining an approximate Gaussian distribution with different covariance parameterizations. We propose that when the approximation is Gaussian, practitioners should favor structured, less flexible covariance parameterizations (with lower degrees of freedom) over unrestricted, highly flexible ones. This approach can lead to better results by capturing more effectively the true correlations that naturally arise in NNs.

Chapter 5

Modeling Weight Correlations in Approximate Inference: When Structure Matters More Than Flexibility

---

Modern deep networks, known for their efficiency in processing high-dimensional data, leverage techniques like parameter sharing [4]. These inherent properties, however, induce correlations between NN weights. This chapter investigates ways of modeling these weight interactions/correlations within the Variational Inference framework, a topic still relevant for developing Bayesian Neural Networks with finely calibrated uncertainties in large models. Supported by related research and empirical evidence, we argue that under the Gaussian distribution assumption, practitioners should focus on imposing less flexible, structured representations rather than computationally expensive, unstructured ones when modeling these correlations. To validate

this proposition, we introduce the Circulant Normal approximate distribution, which inherently captures the aforementioned properties. We evaluate its effectiveness experimentally across diverse tasks. Our method not only outperforms existing approaches on a wide range of benchmarks but also offers lightweight space and time complexity, making it a compelling choice for future research and practical applications in Bayesian deep learning

The rest of this chapter is organized as follows. We begin by discussing the challenges associated with approximate inference distribution modeling and outlining our contributions in Section 5.1. Next, Section 5.2 reviews related work. In Section 5.3, we establish the theoretical background for this chapter and motivate our intuition, that *structure matters more than flexibility* in approximate inference. We then introduce our proposed method, the Circulant Normal distribution, in Section 5.4 to validate our claims. Finally, Section 5.5 presents an experimental evaluation of our method on various tasks.

5.1 Modeling Weight Correlations & Contributions

Bayesian Deep Learning has been put forward as a solution to deterministic NN pathologies, aiming to combine the effectiveness of Deep Learning with the elegance of Bayesian Inference. From the Bayesian viewpoint, model parameters are to be treated as probabilistic, and the notion of *training the network* "replaced" by *computing the posterior distribution* of the parameters after having observed the training data. The Bayesian paradigm allows for well-calibrated predictive distributions, provides a framework to deal with hyperparameter selection, and deals sufficiently with problems related to training-test distribution shift and catastrophic forgetting.

Unfortunately, Bayesian Inference does have its shortcomings: In most practical models – in a non-deep learning context included – it must be treated as Approximate Inference [15]. Also, even the most naive approximations entail considerable computation overhead. For example, Mean-Field Variational Inference in Bayesian deep learning refers to a Gaussian posterior with a diagonal covariance matrix. This simple model already effectively doubles model parameters [174] as we require to save mean and variance values for each model weight.

The above indicates that correlation modeling flexibility (DoF, degree of freedom),

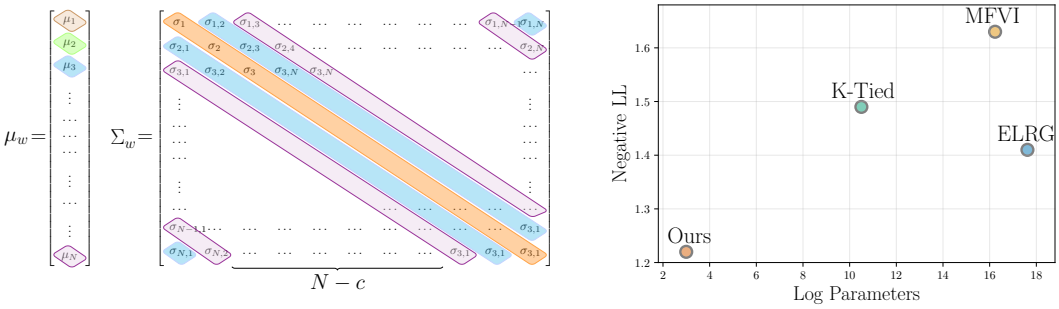

Figure 5.1: Left: Illustration of the parameters (mean μ_w, covariance Σ_w) of the proposed Circulant-Constrained distribution for Variational Inference. Right: Negative Log Likelihood ResNet20 in CIFAR100 vs log parameters of covariance Σ_w per method. Colors indicate shared parameter values. The Circulant Normal allows modeling posterior correlation structure at a negligible computational overhead.

solely does not necessarily translates to better performance. Instead, prioritizing more structured covariance parameterization which account for better correlation modeling between weights can lead to superior results. To validate this, we propose a Circulant-Constrained Gaussian distribution as a family of approximate posterior distributions within the framework of Variational Inference. We take advantage of the fact that Circulant matrices (Figure. 5.1) can be easily parameterized in terms of their generating kernel, as well as easily manipulated by leveraging their Fourier Transform and related eigendecomposition [14]. We show that the proposed model allows for a posterior approximation with a rich correlation structure, an issue where previous works have struggled to deal with effectively. Furthermore, our model succeeds in providing a good trade-off between computational complexity and richness of structure. Numerical experiments on a variety of datasets and benchmarks validate our claims.

5.2 Related Work

We start by exploring methods which leverage the benefits of structured approximate posterior distributions. We then discuss approaches that advocate for simple weight posterior distributions, arguing that neural networks possess inductive biases that can compensate for simpler posterior modeling. Finally, we briefly review methods that

employ Circulant and Toeplitz matrices, first as covariance matrices within Gaussian distribution assumption and then as neural network weight parameterizations.

5.2.1 Correlated Weight Posteriors

Many works try to explain the pathologies of BNNs by exposing the problems of MFVI especially in smaller models [200, 41, 104, 201]. The authors in [103] theoretically proved that MFVI has an additional gap in the ELBO, compared to a suitable, purpose-built posterior. A popular way of overcoming these limitations is to introduce correlations in the approximate posterior. Works such as [12, 202] promote low rank structured correlations, while [203, 204] used the matrix Normal approximate posteriors. Empirically, the introduction of correlations is justified by [11], showing that weights in CNNs are spatially correlated, while authors in [205] showed that correlations can be useful in models that use a form of weight sharing.

5.2.2 Simple Weight Posterior for Deep Neural Networks

In the setting of deep neural networks the approximate posterior distribution of the weights tend to have a simple structure. As the authors in [41] showed, the MFVI posterior can induce similar distributions in function-space to those induced by shallower networks with complex weight posteriors. This idea has been extended by [200] who showed that for deep models there exist MFVI approximate posteriors which provide flexible uncertainty estimates. Empirically, works had successfully applied MFVI approximate posteriors [97, 206, 207]. Closely related to our work, some authors applied even more restrictive approximations; a compact parameterization of the MFVI [13] and a spherical diagonal plus low-rank correlation components [12].

5.2.3 Circulant and Toeplitz Covariance Matrices

These type of covariance matrices arise naturally when data are regularly spaced and the covariance function is stationary. Typical applications can be found in the field of signal processing [208] where the goal is to estimate the empirical covariance using few samples [209]. More related to our work the properties of these could be exploited to accelerate inference for Gaussian Processes (GPs) [210, 211, 212, 213].

5.2.4 Circulant Weight Matrices

Circulant or other low displacement rank neural network weights configurations have been explored in the previous years. Among the first pioneers, authors in [214, 215] focused on restricting the weight matrices to a specific structure to achieve high network compression. Followed by further works [216, 217, 218], who experimented with circulant adaptation of modern neural network components, Notably authors in [219] worked on a theoretical justification of this parameterization. For further applications we refer the reader to the survey of [220].

5.3 Background & Motivation

Following a fully supervised learning setting, where g_w is a Neural Network defined by weights denoted as w. [1] A fully Bayesian solution, if we assume a prior distribution $p(w)$, aims to compute distributions for both the posterior and predictive distribution. Since the posterior cannot be obtained in closed form, we cannot apply exact inference methods. Thus we must resort to approximate inference $p(w|D) \approx q(w)$ where we try to approximate the true posterior distribution. Multiple ways to define q have been put forward. For example, Laplace Approximation involves approximating q as a Gaussian with mean equal to the MAP solution and covariance matrix defined as a function of the loss geometry around the mean [181]. The Stochastic Weight Averaging-Gaussian method (SWAG) approximates posterior parameters as a function of the objective optimization method [10].

In this work, we focus primarily in Variational Inference, as this framework can still provide excellent result in various applications [221, 222]. Variational inference involves defining an approximant in terms of a lower bound on model evidence $p(D)$:

$$p(D) = \mathcal{L}(w) + \mathrm{KL}(q(w)||p(w|D)), \tag{5.1}$$

where $\mathrm{KL}(\cdot)$ stands for the Kullback-Leibler divergence, and $\mathcal{L}(w)$ is the Evidence Lower Bound (ELBO). The ELBO is the objective to be maximized, and can be written as:

$$\mathcal{L}(w) = \mathbb{E}_{q(w)}(p(y|g_w(x^\star))) + \mathrm{KL}(q(w)||p(w)). \tag{5.2}$$

[1] The term "weights" will refer to both weights and biases of a NN, unless stated explicitly. It will be interchangeable with the term "parameters" (of a NN).

The true posterior $p(w|D)$ is far from being unimodal or Gaussian, as an application of Bayes' law, involves a highly non-linear component represented by the Neural Network $g_w(\cdot)$. On the other hand, optimization of the ELBO will require an effective way to compute the expectation $\mathbb{E}_{q(w)}(p(y|g_w(x^\star)))$, and an effective way to compute the KL divergence $\mathrm{KL}(q(w)||p(w))$. Hence, choosing the approximation family for $q(\cdot)$ must be done under these considerations.

Previous work has experimented with various forms for $q(\cdot)$, most notably the Gaussian or other related location-scale parameterized distributions (e.g. Cauchy [123]). In particular, as the covariance parameter scales quadratically with the number of parameters, a number of ways to constrain the form and degrees of freedom (DOFs) of the covariance have been explored [12, 123, 174]. One key insight concerning the choice of the approximate posterior family is that while many works have prioritized a covariance parameterization allowing variation w.r.t. variances of different model weights, empirical evidence seems to suggest that this kind of flexibility is rather left underexploited during learning. Parameter variances tend to converge to values close to prior variance (see Figure 5.2), which is typically chosen to be uniform across variates. This is corroborated by related experiments in recent work; [12] find that the posterior tends towards an isotropic, diagonal structure, with the rank part acting as a regularizer. [13] further criticize shortcomings related to a diagonal covariance structure.

We build on this trend and introduce Circulant Normal a distribution parameterization that sacrifices flexibility (in the aforementioned sense), in favor of allowing richer correlation structure. On this note, the authors in [223] hypothesized that even with simplistic assumptions over a prior, during optimization the parameters will find a version of the network where this restriction is least costly, which our work bears out.

5.4 The Circulant Normal Distribution

We approximate the true posterior as a Gaussian distribution:

$$p(w|D) \approx q(w) = \mathcal{N}(\mu_w, \Sigma_w), \tag{5.3}$$

where $\mu_w \in \mathbb{R}^N$ is the mean of variational posterior distribution and $\Sigma_w \in \mathbb{R}^{N\times N}$ is the positive definite covariance matrix. In any practical setting, the dimensionality of

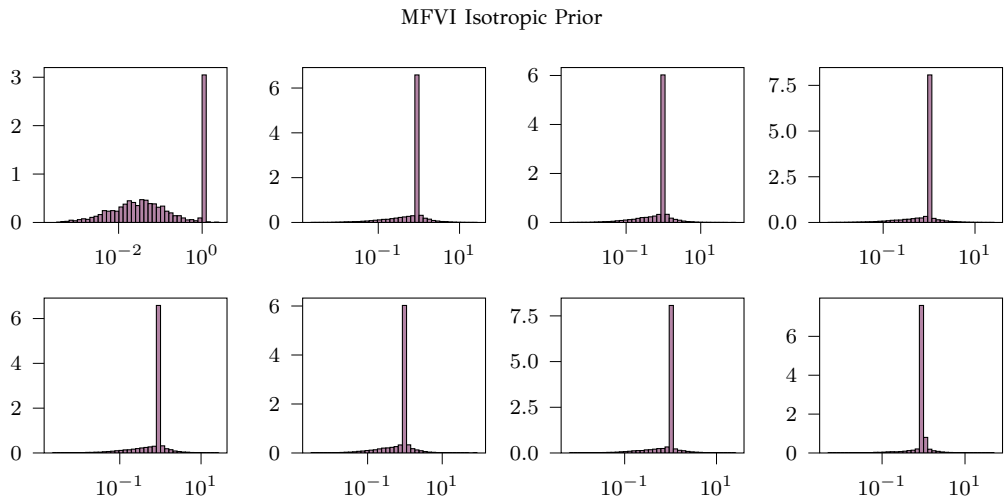

Figure 5.2: Histogram plot of learned diagonal covariance matrix values in Resnet-20, equipped with MFVI posterior, and trained on CIFAR100 under isotropic prior.

w is very high, and modeling an unconstrained Σ_w is infeasible. Hence, we must place some form of restriction over the structure of Σ_w. In this work, we use a circulant covariance-constrained Normal distribution for modeling the weights (cf. Figure 5.1). In this text, we shall refer to this distribution as Circulant-Constrained Gaussian, or Circulant Normal (CN) for brevity.

In order to train our model under this constraint, we will require two basic ingredients: sampling from the variational approximation and computing the KL divergence between the approximate posterior and the prior. These are necessary to optimize the ELBO (eq. 5.2). Concerning sampling, for an (unconstrained) Gaussian posterior we write:

$$w \sim q(w) \equiv w = \mu_w + L\epsilon, \tag{5.4}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and L is such that $LL^T = \Sigma_w$. Usually L is lower triangular, and related with Σ_w through a Cholesky decomposition. In our model, eq. 5.4 is formulated as follows:

$$w \sim q(w) \equiv w = \mu_w + k \circledast \epsilon, \tag{5.5}$$

Where $k \in \mathbb{R}^N$ is a vector which is convolved with white noise ϵ. From the 'kernel' vector k we can define the corresponding circulant matrix C_k where $C_k \in \mathbb{R}^{N \times N}$ and its elements are defined from the kernel k as $C_k(i,j) = k_{(i-j) \mod N}$. The convolutional operation can now be described as a matrix multiplication $k \circledast \epsilon = C_k \epsilon$. This sampling procedure imposes variational covariance matrix $\Sigma_w = C_k C_k^T$.

Lemma 5.4.1. *The product of two circulant matrices is also a circulant matrix.*

Proof. Let C_1 and C_2 be two circulant matrices. The proof takes advantage from the spectral decomposition of circulant matrices. This states columns of the inverse Discrete Fourier Transform (DFT) matrix Q are eigenvectors of any circulant matrix [14], while the corresponding eigenvalues are the DFT values of the kernel generating the circulant matrix, $\Lambda_{C_1} = Q^{-1}k_1$:

$$C_1C_2 = Q\Lambda_{C_1}\cancel{Q^H}\cancel{Q}\Lambda_{C_2}Q^H = Q\Lambda_{C_1}\Lambda_{C_2}Q^H. \tag{5.6}$$

Thus the product is also a circulant matrix, with its kernel computed as the DFT of $diag\{\Lambda_{C_1}\Lambda_{C_2}\}$. □

Following the above lemma we can see that Σ_w is also circulant. However, for Σ_w to be a valid covariance matrix, its positive definitivity must be ensured.

Lemma 5.4.2. *The result of the multiplication of a circulant matrix by its transpose, is positive semi-definite.*

Proof. A matrix is positive semi-definite if it is symmetric and all its eigenvalues λ are non-negative. The matrix C as an outer product is symmetric and its eigenvalues following the previous lemma are $\Lambda_{C_1}\Lambda^{\star}_{C_1}$, where "$\star$" denotes conjugation. This equals $\Lambda^2_{C_1}$, hence all eigenvalues are non-negative. □

Concerning the KL-divergence between the matrix-variate Gaussian posterior and the proposed CN, we have:

$$\begin{aligned}\mathrm{KL}(q(\mu_w, \Sigma_w)||p(\hat{\mu}_w, \hat{\Sigma}_w)) = \frac{1}{2}[\log\frac{|\hat{\Sigma}_w|}{|\Sigma_w|} - N + \\ \mathrm{Tr}(\hat{\Sigma}_w^{-1}\Sigma_w) + (\mu_w - \mu_w)^T\hat{\Sigma}_w^{-1}(\mu_w - \mu_w)]\end{aligned} \tag{5.7}$$

where $\hat{\mu}_w, \hat{\Sigma}_w$ are the prior parameters for the weights w. In order to compute the KL in closed form we have to compute the determinant of the variational circulant covariance where $|\Sigma_w| = \prod_i(\lambda_k)^2$ getting the eigenvalues from Lemma 1 and 2. In order to ensure the numerical stability related to the logarithm and to ensure a strong positive definite covariance matrix, the KL is computed w.r.t. to the closest positive definite matrix to Σ_w, simply by adding a small value to the logarithm computation. As for the matrix trace, involved when the prior covariance is a scaled version of the identity matrix, it can be computed as the sum of the eigenvalues $\mathrm{Tr}(\Sigma_w) = \sum_i(\lambda_k)^2$. If there is another prior matrix, we can simply convolve them using the inverse Fourier Transform.

5.4.1 Exploiting the Circulant Structure

As it is common practice in image and signal processing the only reason for k to be of length N is to be able to perform the multiplication when the circulant is created. Thus the vector k can have $c << N$ learnable parameters (DOFs) while the rest $N - c$ can be zero-padded in order to define the circulant matrix.

Different c values induce different circulant structure in the Σ_w matrix. The zero elements added in k correspond to zero off-diagonals in Σ_w. Thus the c parameter controls both the flexibility and the structural rigidity of our method. At this point it is important to note that even one non-zero off-diagonal in Σ_w can enforce heavy correlations between many weights. In fact, the spherical mean field factorized Gaussian (i.e. diagonal Σ_w) can be seen as a special case of the Circulant Normal where the $c = 1$. In our experiments we found that even a small c can induce heavy correlation in the approximate posterior and yield calibrated predictions.

The authors in [11] showed empirically that weights correlate strongly with neighboring pixels, and anti-correlate or do not correlate with distant ones. This property is fully incorporated in the notion of our zero-pad covariance version, as increasing the zero padding (deceasing c) removes offset diagonals from the main diagonal which correspond to correlations of far spaced weight elements in the weight matrix. In other words the hyperparameter c controls the proximity of neighboring weight pixels up to which we wish to model the correlations. This in addition to the empirical property from [11] allows us in practice to set c to take values even up to 20 (see experiments) and still get excellent results as this models the between kernel correlations (mainly 3×3 in modern CNNs) and disabling the cross channel correlations which do not contribute too much.

5.4.2 Concerning the Prior Distribution

As in any Bayesian formulation, we will need a distribution that will encode our belief about the inferred variables prior to considering our observations [224]. A standard Normal distribution states that all weights are distributed according to the same variance, with zero correlations. Formally we write $p(w) = \mathcal{N}(0, I)$. Another option is to place a prior that will encode circulant-related correlations explicitly. The standard Normal distribution is a good choice to encode an agnostic prior belief, apart from that weights are more likely to be distributed around zero. An identity covariance

is indeed a circulant matrix itself, however it is rather an edge case of the circulant family due to its only (repeating) unity eigenvalue, which leads to trivial circulant structure and correlation. In the context of the stated motivation for using circulant matrices, we may want to allow weights to be more likely to be explicitly correlated. To this end, we consider a circulant prior with a random kernel, so formally we have $p(w) = \mathcal{N}(0, \Sigma_k)$ with $k \sim \mathcal{N}(0, I^c)$. Matrix I^c denotes a diagonal matrix where indices $i \leq c$ are equal to unity, and the rest are set to zero. The hyperparameter c effectively controls the amount of structural penalty enforced to the posterior; for example, $c = 1$ will imply a spherical, uncorrelated prior.

Furthermore, instead of choosing the kernel values at random the circulant prior can compute the values via empirical Bayes in closed analytical form [225, 226]. where assuming that the approximate distribution is a Gaussian distribution. We use $p(w) = \mathcal{N}(0, \Sigma_k)$, where Σ_k is generated by a kernel formed as $k = [k_0 k_1 \cdots k_N]$, Its values are given by:

$$k_n = \text{Tr}(P^n \Sigma_w) + \mu_w^T P^n \mu_w, \tag{5.8}$$

where P^n is the n^{th} power of the permutation matrix P. P is defined as the square matrix that permutes row n to row $n + 1$ mod k, when it multiplies a vector from the left. A full mathematical derivation of the empirical circulant distribution can be found in Appendix C

5.5 Experimental Evaluation

This section presents numerical results to evaluate the benefits of the circulant Normal. We begin by quantifying the advantages of CN distribution as an approximate posterior in various variational inference settings (Section 5.5.1). Next, Section 5.5.2 explores methods for integrating CN distributions into existing state-of-the-art VI methods to further enhance their predictive performance. Finally, Section 5.5.3 investigates the benefits of using structured CN distributions as priors to enforce more correlated approximate posteriors.

Throughout our experiments, we found that the proposed method provides good predictive uncertainties while having excellent computational efficiency on a variety of settings, highlighting the benefits of Bayesian inference with a circulant-imposed structure on the posterior. Concerning implementation details of the proposed and

compared models, and benchmark setup in general, we have moved additional information to the Appendix C. In all cases, we treat inference on a per-layer basis (so in practice the posterior covariance is block-circulant).

5.5.1 Evaluating Approximate Circulant Posteriors

We compared our method against several covariance approximation schemes from the literature. This included the popular mean field approximation (mean field Variational Inference, MFVI) [15], which assumes a diagonal posterior covariance matrix. We further included a compact parameterization of the diagonal covariance [13] (k-Tied) and the low-rank extension of the diagonal covariance [12] (ELRG). Throughout our experiments, we found that the proposed method provides good predictive uncertainties while having excellent computational efficiency and order of magnitude less parameters, on a variety of settings.

UCI Regression

We begin by testing our method on the UCI regression tasks [196]. We experiment with 8 UCI regression datasets using standard training-evaluation-test splits from [63] and their GAP-variants [195], which are specifically designed to test "in-between" uncertainty (i.e. the ability of the model to predict high uncertainty values as test data move away from the observed data). In this test, we use a fully connected architecture with hidden layers that have $[100, 50, 20]$ neurons respectively. The kernel size c for the circulant approximate covariance matrix per layer was set to 20 (equal to only 1.17% of the number of MFVI parameters). To measure performance we deployed Gaussian test log-likelihood (LL), a metric which evaluates both uncertainty and accuracy. The main results are depicted in Figure 5.3. As we can see, the circulant posterior achieves better test log-likelihood compared to other covariance approximation methods. The excellent performance of our method is further highlighted in the GAP-splits, as it yields better LL values in 7 out of 8 datasets considered.

Image Classification Under Distribution Shift

Next, we evaluate our method on standard image classification tasks over the CIFAR10, CIFAR100 datasets where we employed a ResNet-20 [20] [197]. A robust Bayesian inference technique is particularly important in deep learning applications,

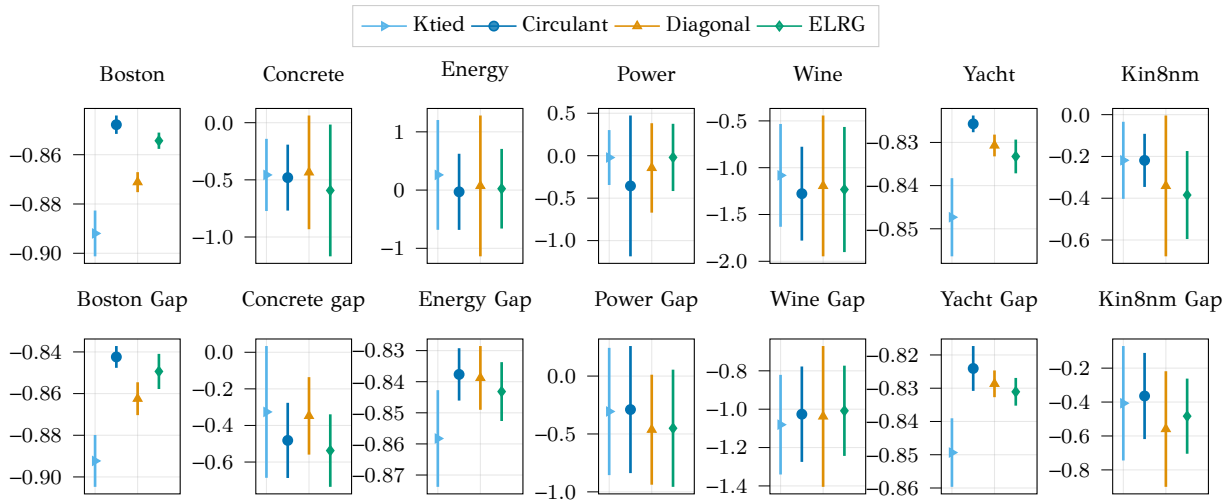

Figure 5.3: Numerical results for regression trials on UCI datasets [196]. Mean values of test Log-Likelihood (↑) are shown with ± 1 standard deviation error bars, obtained over standard [63] and GAP [195] splits.

as these models often exhibit suboptimal calibration [32]. We also provide experiments in a context of high degree of distribution shift, as under these conditions the evaluation of predictive uncertainty is the most useful in practice [193]. The kernel size c for the circulant approximate covariance matrix per layer was set to 20 (equal to only 0.003% of the number of MFVI parameters) the additional overhead that each method introduces in order to compute the covariance matrix (in the Resnet-20 case) is depicted in table 5.2.

According to Table 5.1 our proposed approximation outperforms the other methods in terms of log-likelihood and expected calibration in both standard and corrupted data, while still having high competitive performance in terms of error and Brier score. Furthermore it produces predictive uncertainty estimates that enable successful semantic shift detection, as the AUROC values demonstrate. Overall, these results suggest that the proposed approach produces better calibrated and more accurate models than other popular uncertainty quantification approaches.

Transfer Learning Experiments

Seeking a more rigorous evaluation of our method we moved away from the standard benchmark datasets and small models towards a more "real-world" setting scenario. We followed the experimental protocol of [227, 80] to evaluate our claims on a transfer

Table 5.1: Numerical results for classification trials on Corrupted CIFAR100/CIFAR10 datasets. Log-Likelihood , Expected Calibration Error and Error. The area under the ROC (AUROC) of a for binary classifier using the predictive entropy values to distinguish CIFAR (in-distribution) from SVHN (out-of-distribution) examples.

Dataset	Method	Standard			Corrupted			OOD
		Error ↓	LL ↑	ECE ↓	Error ↓	LL ↑	ECE ↓	AUROC ↑
CIFAR100	MAP	0.32 ±0.09	−3.47 ±1.07	0.26±0.01	0.53 ±0.00	−6.89 ±0.07	0.42±0.00	0.74 ±0.02
	MFVI	0.32 ±0.00	−1.63 ±0.47	0.14±0.00	0.57 ±0.00	−3.73 ±0.08	0.27±0.00	0.71 ±0.02
	K-Tied	0.35 ±0.02	−1.49 ±0.25	0.07±0.06	0.57 ±0.02	−3.30 ±0.75	0.20±0.08	0.73 ±0.02
	ELRG	**0.29** ±0.00	−1.41 ±0.16	0.11±0.01	**0.52** ±0.00	−3.38 ±0.04	0.24±0.01	0.72 ±0.03
	Circulant	0.31 ±0.00	**−1.22** ±0.01	**0.06**±0.00	0.55 ±0.00	**−2.95** ±0.07	**0.18**±0.00	**0.76** ±0.02
CIFAR10	MAP	**0.06** ±0.00	−0.54 ±0.01	0.04±0.00	**0.19** ±0.02	−2.08 ±0.39	0.16±0.02	0.89 ±0.02
	MFVI	**0.06** ±0.00	**−0.25** ±0.00	**0.02**±0.00	0.22 ±0.02	−1.11 ±0.29	0.12±0.01	0.87 ±0.01
	K-Tied	0.07 ±0.00	−0.31 ±0.01	0.03±0.00	0.22 ±0.03	−1.24 ±0.29	0.13±0.01	0.88 ±0.01
	ELRG	**0.06** ±0.00	−0.26 ±0.05	**0.02**±0.00	**0.19** ±0.02	−1.00 ±0.31	**0.10**±0.03	**0.89** ±0.01
	Circulant	0.08 ±0.01	−0.27 ±0.01	**0.02**±0.00	0.23 ±0.01	**−0.89** ±0.04	**0.11**±0.01	**0.89** ±0.00

Table 5.2: Total number of variational parameters for the covariance matrix computation, alongside the computational time in seconds for one forward and backward pass for each method on ResNet-20.

Method	Parameters	Forward	Backward
MFVI	11211328	0.016±0.00	0.123±0.00
K-Tied *(k=3)*	109245	0.017±0.00	0.059±0.00
K-Tied *(k=1)*	36415	0.016±0.00	0.035±0.00
ELRG *(k=2)*	44841728	0.016±0.00	0.035±0.00
ELRG *(k=1)*	22420864	0.016±0.00	0.034±0.00
Circulant *(c=20)*	**420**	0.016±0.00	0.031±0.00

learning problem. We considered three additional real-world large scale datasets: the APTOS Blindness Detection[228], Melanoma Classification [229], and Cassava Leaf Disease Classification [230]. We used an ImageNet-pretrained ResNet-50 and fine-tuned it on each dataset for 30 epochs. In Table 5.3, we find that the circulant posterior often outperforms the baselines while achieving significant improvement in uncertainty quantification. This is a strong indication that our methods can generalize beyond small scale datasets and models.

Table 5.3: Numerical results for transfer learning trials on 'real-world' datasets. Log-Likelihood, Expected Calibration Error, Brier Score, Error and Selective prediction accuracy are used for comparison

Dataset	Method	Error ↓	LL ↑	ECE ↓	Sel. ↑
APTOS	MFVI	0.26	−0.79	0.07	89.7
	ELRG	0.27	−0.82	0.06	88.1
	Circ.	**0.25**	**−0.78**	**0.04**	**89.8**
CASSAVA	MFVI	**0.20**	**−0.56**	**0.01**	93.70
	ELRG	**0.20**	−0.60	0.05	**93.74**
	Circ	**0.20**	−0.58	**0.01**	93.08
MELANOMA	MFVI	**0.01**	−0.084	**0.01**	99.17
	ELRG	**0.01**	−0.079	**0.01**	99.25
	Circ	**0.01**	**−0.077**	**0.01**	**99.27**

Beyond Isotropic Priors

Up to this point, in our experiments we tested the benefits of a circulant approximate posterior distribution but in a VI setting where the prior was set to be an isotropic Gaussian. The choice of prior distribution could potentially affect the posterior performance and the overall stochastic optimization general. We tried to investigate how our proposed posterior behaves when the prior distribution changes significantly. We thus deployed two additional priors. First we used the multivariate Gaussian with Matérn covariance from [11] where the prior for all the convolutional layers was set to be block diagonal where each block a toeplitz matrix specially crafted by the authors to model correlations within convolutional filters. Also we adopted the induced functional prior from [231] where authors assume a Gaussian prior over the model parameters with parameter values that minimize the Wasserstein distance between BNN induced functions and the ones acquired from a target Gaussian process model. Numerical results are showed in Table 5.4 indicate that the proposed can produce valuable uncertainty estimates especially under semantic scenario.

5.5.2 Circulant Structure as Improvement to Existing VI Methods

Furthermore, we carried out experiments to highlight some additional usages of the circulant normal. We argue that the proposed CN distribution can be used as a "plug-and-play" replacement of the diagonal normal in current state-fo-the-art methods to further boost their performance Specifically, we start by highlighting that the circu-

Table 5.4: Numerical results for Renset-20 on CIFAR100 for different prior and posterior combinations. Log-Likelihood, Expected Calibration Error and the area under the ROC (AUROC) are used for comparison. *GPI* indicates the GP-induced Gaussian priors from [231] and *MVG* Matrix variate Gaussian Matérn covariance from [11].

Prior	Posterior	Standard		Corrupted		OOD
		LL ↑	ECE ↓	LL ↑	ECE ↓	AUROC ↑
MVG	MFVI	−3.43	0.26	−6.80	0.429	0.742
	ELRG	**−3.36**	**0.26**	**−6.18**	**0.411**	0.736
	Circ.	−3.72	0.27	−6.61	0.423	**0.743**
GPI	MFVI	**−3.21**	**0.24**	−6.36	**0.40**	0.72
	ELRG	−3.31	0.26	**−5.92**	0.41	0.72
	Circ	−3.62	0.26	−6.68	0.41	**0.77**

Table 5.5: Numerical experiments for ResNet-20 with rank1 parameterization [123] trained on CIFAR100. *Diag mix k=3* indicates vanilla approximate posterior (mixture of 3 Normal distributions), *Circ. mix k=3* mixture of 3 circulant Normal and *circ mix k=6* mixture of 6 ciculant Normal distributions.

Posterior	Standard		Corrupted	
	LL ↑	ECE ↓	LL ↑	ECE ↓
Diag mix k=3	−3.23±0.10	**0.24**±0.00	−6.77±0.25	0.41±0.00
Circ. mix k=3	−3.26±0.08	**0.24**±0.00	−6.53±0.01	0.40±0.00
Circ. mix k=6	**−3.10**±0.07	**0.24**±0.00	**−5.89**±0.05	**0.38**±0.00

lant normal can be a part of more complex approximate posterior for modeling the neural network weights. We use as a baseline the popular rank-1 parameterization of [123], where we replaced the original mean field Gaussian mixture components with circulant ones of kernel size $c = 20$. This allows us to enjoy the benefits of a mixture of structured distributions while having reduced the parameters compared to the vanilla mixture. The dramatic decrease in additional parameters for the circulant mixture allows us in practice to double the number of components in the approximate posterior. The numerical results can be found in Table 5.5, where the mixture approximate posterior distribution comprised circulant components yields better calibrated prediction for both in and out distribution data, while the extra mixture components (Circ. mix k=6) (which came as a result of using the circulant approximation) significantly boost the overall performance.

Table 5.6: Numerical experiments for ResNet-20 with ELRG [12] approximate posterior trained on CIFAR100. *Diagonal* indicates vanilla isotropic prior, *Rand Circ.* circulant prior with random valued kernel *Empir Circ.* Empirical Circulant.

Prior	Standard		Corrupted	
	LL ↑	ECE ↓	LL ↑	ECE ↓
Diagonal	$\mathbf{-1.27}_{\pm 0.00}$	$0.10_{\pm 0.00}$	$-3.07_{\pm 0.04}$	$0.22_{\pm 0.01}$
Rand Circ.	$-1.46_{\pm 0.06}$	$0.06_{\pm 0.01}$	$-2.76_{\pm 0.13}$	$0.17_{\pm 0.01}$
Empir Circ.	$-1.35_{\pm 0.02}$	$\mathbf{0.05}_{\pm 0.00}$	$\mathbf{-2.54}_{\pm 0.09}$	$\mathbf{0.14}_{\pm 0.00}$

5.5.3 Evaluating Effectiveness of Circulant Priors

Also we wanted to test whether our method can function as a prior for structured approximate posterior distributions. We adopted the ELRG [12] posterior approximation method but we replaced the spherical Gaussian prior with a circulant one with kernel size was set to $c = 20$. We experimented tested both random and empirical circulant priors, with two different versions of circulant priors, one where the prior kernel values where initialized based on the Normal distribution and another where the values where computed using empirical Bayes. In Table 5.6, we can observe their performance.

The circulant empirical prior and even the one with the random kernel values is able to produce higher quality predictive samples via inducing further correlations to the approximate distribution. The structured prior achieves excellent performance especially in corrupted test under covariance shift. As we can see from the results in Figure 5.4 (see Appendix for further visualizations) the circulant prior enforces more non-diagonal elements of the covariance matrix to be non-zero, versus the plain diagonal prior.

5.5.4 How Circulant Kernel Size Affects Predictive Performance

We carry out experiments in order to study the behavior of our method under different kernel size c choices, which affect the number of non-zero off-diagonal terms in the covariance matrix. In Figure 5.5, we quantify the performance benefits of linearly increasing the kernel size. We can see clearly that increasing the kernel size of the approximate posterior is strongly coupled with numerical gain. More structured circulant covariance matrices yield increasingly calibrated results as the performance on

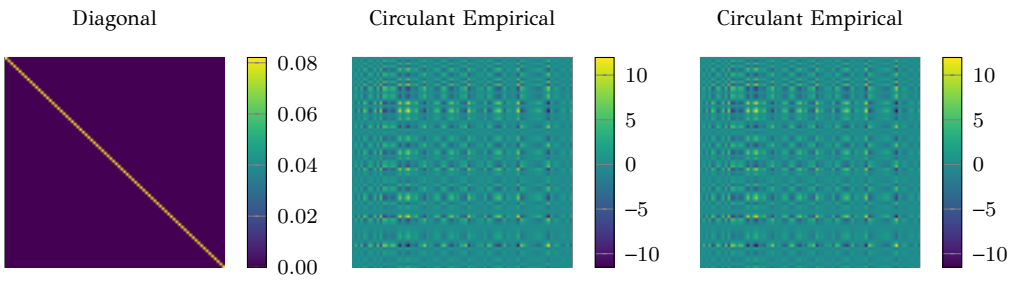

Figure 5.4: Illustration of learned covariance matrices for the $1st$ convolutional layer of Resnet-20 trained on CIFAR100 for diagonal prior and for empirical circulant prior distribution.

out-of-distribution data suggests. Additionally, in Figure 5.7 we plot the corresponding covariance matrices for the first convolutional layer of ResNet-20 for increasing kernel size. As the the c increases, offset diagonal rows are introduced to the approximation. Furthermore we can observe that the as the kernel size increases the magnitude of the variance values increases as well. In Figure 5.6, we can see how the kernel size impacts the computational time of our method. While the time for forward pass and the computation of the KL divergence seems to be independent of the kernel size, the time for computing gradients (backward pass) raises exponentially as c is increased. In our experiments we found that there is a limit at $c = 2000$ which after that the training overhead becomes too big to justify the benefits of a larger circulant kernel.

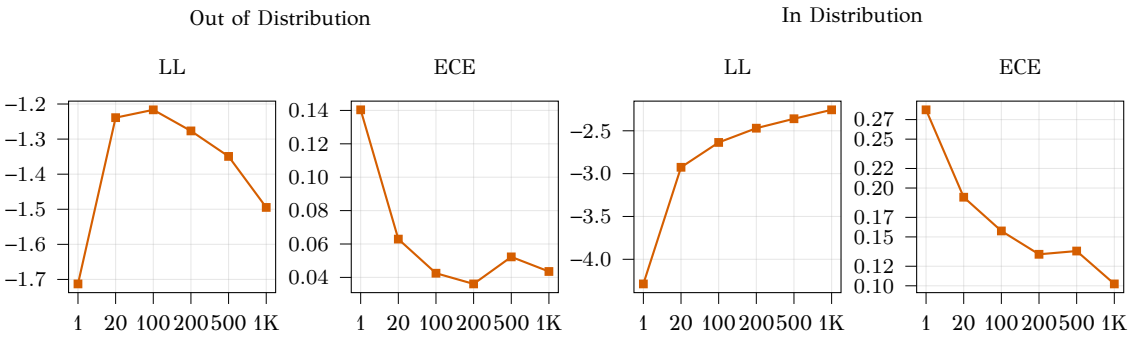

Figure 5.5: Numerical results for classification trials on CIFAR100. X-axis correspond to increasing kernel size for the circulant approximate posterior.

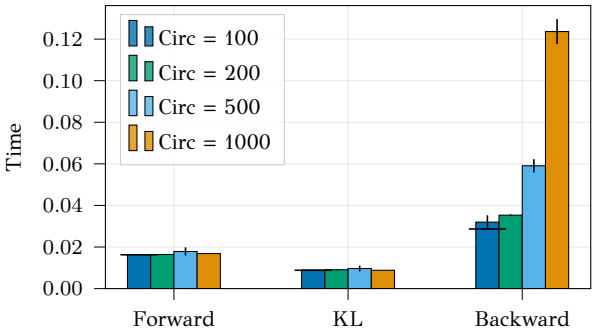

Figure 5.6: Computational complexity for the circulant method against other covariance parameterizations (top), time for increasing kernel size (bottom). *Forward* indicates the time for a single forward pass, *KL* is the time for computing the Kullback–Leibler term in the ELBO loss and *Backward* indicates the time for computing the gradients w.r.t. posterior parameters.

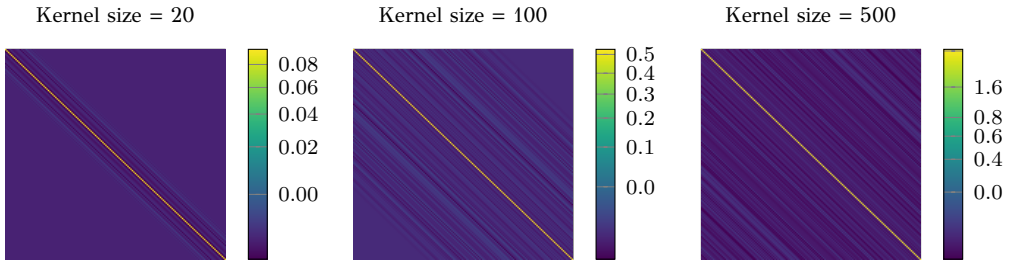

Figure 5.7: Illustration of learned covariance matrices for the first convolutional layer of Resnet-20 trained on CIFAR100 for increasing circulant kernel size.

5.6 Discussion

In this work, we explore and evaluate the idea that introduction of correlations within the prior and posterior distribution had a more significant impact on the model's uncertainty estimates than simply increasing its flexibility. We validate our claims by introducing the Circulant Normal distribution for approximate inference, which leverages a circulant structure for the covariance matrix. This unique structure offers attractive mathematical properties while enabling the incorporation of interpretable correlation patterns inspired by existing literature into the network weights. Remarkably, this approach allows us to drastically reduce the posterior's flexibility (number of parameters) while effectively maintaining a substantial level of correlations. Our experiments demonstrated, across diverse scenarios, that the Circulant Normal distribution effectively serves both as a posterior and as a prior for NN weights.

Chapter 6

Conclusions and Future Work

This chapter concludes the dissertation by summarizing our main contributions to efficient Bayesian deep learning across the applications considered throughout this thesis. We then in Section 6.2 discuss future research directions and potential applications for each individual contribution.

6.1 Summary of Contributions

In Chapter 3, we proposed a novel probabilistic neural network module, the Variational Feature Pyramid Network (VarFPN), specifically designed for robust and probabilistic object detection. VarFPN builds upon the widely used Feature Pyramid Network (FPN), a feature fusion module frequent in object detection architectures. Our method offers several benefits due to its stochastic architecture. We introduce probabilistic architecture parameters and VarFPN learns these parameters via variational inference. The fusion architecture can adapt to the training data, leading to a customized feature fusion process for each dataset. Additionally, the probabilistic design allows for uncertainty estimates to be incorporated into the final object detection prediction. These uncertainty estimates are of good quality despite the extremely

low number of probabilistic parameters, due to the significant impact these parameters have on the final detection results. We tested VarFPN with various detectors on both detection and segmentation tasks. It consistently outperformed related methods, highlighting the twofold benefits of our approach.

In Chapter 4, we focused on the challenges of applying Bayesian inference to large neural networks. The high dimensionality of probabilistic parameters is a significant hurdle in Bayesian modeling, as these methods notoriously scale poorly. Low-dimensional Bayesian deep learning offers a solution. Like the previous chapter, where a small number of neural network parameters is treated probabilistically, allowing for more flexible and multimodal posteriors within a lower-dimensional space. We introduced *INR Inference*, which involves learning an efficient hypernetwork that utilizes elements from implicit neural representation methods. This hypernetwork generates probabilistic variables that introduce stochasticity into the weight parameters of the main neural network. Since Bayesian inference is performed over the low-dimensional space of hypernetwork parameters, enables richer approximations for the desired posterior distribution. This advantage of INR Inference is qualitatively evaluated in various experiments. Our results suggest that, for practitioners, it may be more effective to introduce flexible/multimodal stochasticity into the neural network weights rather than attempting to poorly approximate them directly.

We continued in Chapter 5, where we studied the problem of approximate posterior modeling in BNNs within the variational inference framework. Variational inference requires specifying an approximate posterior distribution. Ideal posteriors are easy to optimize, with analytical or tractable properties, while also being flexible enough to theoretically capture the true posterior (often assumed to be non-linear). In our work, we challenged the emphasis on pure flexibility in favor of structured posteriors that leverage correlation modeling with a restricted number of degrees of freedom (DoF). To validate our claims, we introduced the Circulant Normal distribution. This is a structured parameterization of the Gaussian that offers efficiency and structure at the same time by restricting the DoF. Empirical evidence across a wide range of benchmarks demonstrates that our approach outperforms alternative methods by combining our flexible parameterization with low space and time complexity.

6.2 Future Research Directions

Our proposed VarFPN method holds promise for further exploration. One promising direction involves investigating other forms of initial complex architectures, incorporating more hidden layers and modules. We also plan to experiment with different sparse prior distributions, enabling the exploration of more complex variational distributions over the model weights. We envision employing a Student's-t distribution using a Gaussian-Gamma factorization, while other options like the Weibull or Gamma distributions could be considered to constrain weights as strictly positive, following the works [232] or [233]. Additionally, a valuable future direction would be to examine whether posterior collapse [138] occurs, evaluate if the model converges to a single architecture during inference. We can then explore ways to encourage the model to explore multiple stochastic architectures at test time, leading to more diverse predictions.

There are several promising avenues for further research on the proposed INR inference method. One direction is to explore integrating different INR architectures, such as multiplicative filter networks [234], with our current method. Additionally, investigating the use of alternative approximation techniques, such as Hamiltonian Monte Carlo [36], could be beneficial. To enhance the flexibility of the INR, we could consider adopting learnable positional encodings to augment the input indices, as demonstrated in recent work [235, 236]. Furthermore, developing a proper initialization strategy for the INR hypernetwork that avoids disrupting the main network's existing MAP solution is a promising approach with the potential to significantly reduce training time.

Our future work aims to further explore the potential of low-rank displacement covariance parameterizations for the Circulant Normal distribution. Additionally, we plan a comprehensive analysis of the phenomenon where approximate posteriors tend towards solutions with low flexibility. This exploration will extend beyond the Circulant Normal, examining broader possibilities for structured modeling. Investigating whether the benefits of using restricted structured modeling generalize to other variational inference frameworks. This includes natural gradient methods [237] and in Laplace methods [9]

Bibliography

[1] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 521, 2017, pp. 2117–2125.

[2] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 7462–7473, 2020.

[3] D. Hall, F. Dayoub, J. Skinner, H. Zhang, D. Miller, P. Corke, G. Carneiro, A. Angelova, and N. Sünderhauf, "Probabilistic Object Detection: Definition and Evaluation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 1031–1040.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and Efficient Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 781–10 790.

[6] G. Franchi, O. Laurent, M. Leguéry, A. Bursuc, A. Pilzer, and A. Yao, "Make Me a Bnn: A Simple Strategy for Estimating Bayesian Uncertainty from Pre-trained Models," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 12 194–12 204.

[7] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding Box Regression with Uncertainty for Accurate Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2888–2897.

[8] M. Betancourt, "A Conceptual Introduction to Hamiltonian Monte Carlo," *arXiv preprint arXiv:1701.02434*, 2017.

[9] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig, "Laplace Redux-Effortless Bayesian Deep Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 20 089–20 103, 2021.

[10] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A Simple Baseline for Bayesian Uncertainty in Deep Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 13 153–13 164, 2019.

[11] V. Fortuin, A. Garriga-Alonso, S. W. Ober, F. Wenzel, G. Ratsch, R. E. Turner, M. van der Wilk, and L. Aitchison, "Bayesian Neural Network Priors Revisited," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[12] M. Tomczak, S. Swaroop, and R. Turner, "Efficient Low Rank Gaussian Variational Inference for Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 4610–4622, 2020.

[13] J. Swiatkowski, K. Roth, B. Veeling, L. Tran, J. Dillon, J. Snoek, S. Mandt, T. Salimans, R. Jenatton, and S. Nowozin, "The k-tied Normal Distribution: A Compact Parameterization of Gaussian Mean Field Posteriors in Bayesian Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 119, 2020, pp. 9289–9299.

[14] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc., 1989.

[15] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4, no. 4.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[18] H. Gholamalinezhad and H. Khosravi, "Pooling Methods in Deep Neural Networks, A Review," *arXiv preprint arXiv:2009.07485*, 2020.

[19] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 448–456.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[22] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in Vision: A Survey," *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1–41, 2022.

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] A. Graves, "Generating Sequences with Recurrent Neural Networks," *arXiv preprint arXiv:1308.0850*, 2013.

[25] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding Deep Learning (still) Requires Rethinking Generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[26] D. J. MacKay, "The Evidence Framework Applied to Classification Networks," *Neural computation*, vol. 4, no. 5, pp. 720–736, 1992.

[27] A. G. Wilson and P. Izmailov, "Bayesian Deep Learning and a Probabilistic Perspective of Generalization," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 4697–4708, 2020.

[28] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[29] M. Tipping, "The Relevance Vector Machine," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 12, 1999.

[30] D. J. C. Mackay, *Bayesian Methods for Adaptive Models*. California Institute of Technology, 1992.

[31] B. Kronheim, M. P. Kuchera, H. B. Prosper, and A. Karbo, "Bayesian Neural Networks for Fast SUSY Predictions," *Physics Letters B*, vol. 813, p. 136041, 2021.

[32] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 1321–1330.

[33] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[34] A. Kristiadi, M. Hein, and P. Hennig, "Being Bayesian, Even Just a Bit, Fixes Overconfidence in Relu Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 5436–5446.

[35] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[36] R. M. Neal *et al.*, "MCMC Using Hamiltonian Dynamics," *arXiv preprint arXiv:1206.1901*, 2012.

[37] T. Chen, E. Fox, and C. Guestrin, "Stochastic Gradient Hamiltonian Monte Carlo," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2014, pp. 1683–1691.

[38] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson, "What are Bayesian Neural Network Posteriors Really Like?" in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 4629–4640.

[39] A. Kristiadi, A. Immer, R. Eschenhagen, and V. Fortuin, "Promises and Pitfalls of the Linearized Laplace in Bayesian Optimization," *Fifth Symposium on Advances in Approximate Bayesian Inference*, 2023.

[40] J. Martens, "New Insights and Perspectives on the Natural Gradient Method," *Journal of Machine Learning Research*, vol. 21, no. 146, pp. 1–76, 2020.

[41] S. Farquhar, L. Smith, and Y. Gal, "Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 4346–4357, 2020.

[42] A. Botev, H. Ritter, and D. Barber, "Practical Gauss-Newton Optimisation for Deep Learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 557–565.

[43] H. Ritter, A. Botev, and D. Barber, "A Scalable Laplace Approximation for Neural Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, vol. 6, 2018.

[44] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*. Chapman and Hall/CRC, 2018.

[45] J. Lee, M. Humt, J. Feng, and R. Triebel, "Estimating Model Uncertainty of Neural Networks in Sparse Information Form," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 5702–5713.

[46] W. J. Maddox, G. Benton, and A. G. Wilson, "Rethinking Parameter Counting in Deep Models: Effective Dimensionality Revisited," *arXiv preprint arXiv:2003.02139*, 2020.

[47] F. Bergamin, P. Moreno-Muñoz, S. Hauberg, and G. Arvanitidis, "Riemannian Laplace Approximations for Bayesian Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.

[48] J. Antoran, S. Padhy, R. Barbano, E. Nalisnick, D. Janz, and J. M. Hernández-Lobato, "Sampling-based Inference for Large Linear Models, with Application to Linearised Laplace," in *Fifth Symposium on Advances in Approximate Bayesian Inference - Fast Track*, 2023.

[49] M. Stephan, M. D. Hoffman, D. M. Blei *et al.*, "Stochastic gradient descent as approximate bayesian inference," *Journal of Machine Learning Research (JMLR)*, vol. 18, no. 134, pp. 1–35, 2017.

[50] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging Weights Leads to Wider Optima and Better Generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[51] A. Malinin and M. Gales, "Predictive Uncertainty Estimation via Prior Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[52] B. Charpentier, D. Zügner, and S. Günnemann, "Posterior Network: Uncertainty Estimation Without OOD Samples via Density-based Pseudo-counts," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1356–1367, 2020.

[53] L. K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[54] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[55] F. Seligmann, P. Becker, M. Volpp, and G. Neumann, "Beyond Deep Ensembles: A Large-Scale Evaluation of Bayesian Deep Learning under Distribution Shift," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.

[56] L. Hoffmann and C. Elster, "Deep Ensembles from a Bayesian Perspective," *arXiv preprint arXiv:2105.13283*, 2021.

[57] F. D'Angelo and V. Fortuin, "Repulsive Deep Ensembles are Bayesian," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 3451–3465, 2021.

[58] C. Shui, A. S. Mozafari, J. Marek, I. Hedhli, and C. Gagné, "Diversity Regularization in Deep Ensembles," *arXiv preprint arXiv:1802.07881*, 2018.

[59] T. Abe, E. K. Buchanan, G. Pleiss, and J. P. Cunningham, "Pathologies of Predictive Diversity in Deep Ensembles," *Transactions on Machine Learning Research*, 2024.

[60] D. J. Spiegelhalter and S. L. Lauritzen, "Sequential Updating of Conditional Probabilities on Directed Graphical Structures," *Networks*, vol. 20, no. 5, pp. 579–605, 1990.

[61] D. Barber and C. M. Bishop, "Ensemble Learning in Bayesian Neural Networks," *Nato ASI Series F Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.

[62] A. Immer, M. Korzepa, and M. Bauer, "Improving Predictions of Bayesian Neural Nets via Local Linearization," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2021, pp. 703–711.

[63] J. M. Hernández-Lobato and R. Adams, "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 1861–1869.

[64] T. P. Minka, "Expectation propagation for approximate bayesian inference," *arXiv preprint arXiv:1301.2294*, 2013.

[65] B. J. Frey and G. E. Hinton, "Variational Learning in Nonlinear Gaussian Belief Networks," *Neural Computation*, vol. 11, no. 1, pp. 193–213, 1999.

[66] O. Wright, Y. Nakahira, and J. M. Moura, "An Analytic Solution to Covariance Propagation in Neural Networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2024, pp. 4087–4095.

[67] A. Shekhovtsov and B. Flach, "Feed-forward Propagation in Probabilistic Neural Networks with Categorical and Max Layers," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[68] M. Haußmann, F. A. Hamprecht, and M. Kandemir, "Sampling-free Variational Inference of Bayesian Neural Networks by Variance Backpropagation," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR, 2020, pp. 563–573.

[69] A. Korattikara Balan, V. Rathod, K. P. Murphy, and M. Welling, "Bayesian Dark Knowledge," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[70] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.

[71] A. Malinin, B. Mlodozeniec, and M. Gales, "Ensemble Distribution Distillation," *arXiv preprint arXiv:1905.00076*, 2019.

[72] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems (NeurIPS)*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017.

[73] R. Muthukumar and J. Sulam, "Sparsity-aware Generalization Theory for Deep Neural Networks," in *International Conference on Learning Theory (COLT)*. PMLR, 2023, pp. 5311–5342.

[74] S. Liu, "Learning Sparse Neural Networks for Better Generalization," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 5190–5191.

[75] E. Nalisnick and P. Smyth, "Learning Priors for Invariance," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2018, pp. 366–375.

[76] T. Pearce, A. Y. Foong, and A. Brintrup, "Structured Weight Priors for Convolutional Neural Networks," *arXiv preprint arXiv:2007.14235*, 2020.

[77] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani, "Gaussian Process Behaviour in Wide Deep Neural Networks," *Proceedings of the International Conference on Learning Representations (ICLR)*, vol. 6, 2018.

[78] T. Pearce, R. Tsuchida, M. Zaki, A. Brintrup, and A. Neely, "Expressive Priors in Bayesian Neural Networks: Kernel Combinations and Periodic Functions," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR, 2020, pp. 134–144.

[79] S. Sun, G. Zhang, J. Shi, and R. Grosse, "Functional variational bayesian neural networks," *Proceedings of the International Conference on Learning Representations (ICLR)*, vol. 7, 2019.

[80] T. G. Rudner, S. Kapoor, S. Qiu, and A. G. Wilson, "Function-Space Regularization in Neural Networks: A Probabilistic Perspective," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2023, pp. 29 275–29 290.

[81] B.-H. Tran, S. Rossi, D. Milios, and M. Filippone, "All You Need is a Good Functional Prior for Bayesian Deep Learning," *Journal of Machine Learning Research (JMLR)*, vol. 23, no. 74, pp. 1–56, 2022.

[82] E. Nalisnick, J. Gordon, and J. M. Hernández-Lobato, "Predictive Complexity Priors," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2021, pp. 694–702.

[83] N. Pawlowski, A. Brock, M. C. Lee, M. Rajchl, and B. Glocker, "Implicit Weight Uncertainty in Neural Networks," *arXiv preprint arXiv:1711.01297*, 2017.

[84] A. Graves, "Practical Variational Inference for Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 24, 2011.

[85] M. D. Hoffman and M. J. Johnson, "Elbo Surgery: Yet Another Way to Carve up the Variational Evidence Lower Bound," 2016.

[86] R. J. Williams, "Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[87] R. Ranganath, S. Gerrish, and D. Blei, "Black Box Variational Inference," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2014, pp. 814–822.

[88] G. Casella and C. P. Robert, "Rao-Blackwellisation of Sampling Schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, 1996.

[89] S. M. Ross, *Introduction to Probability Models*. Academic press, 2014.

[90] D. P. Kingma and M. Welling, "Auto-encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[91] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2014, pp. 1278–1286.

[92] M. Jankowiak and F. Obermeyer, "Pathwise Derivatives Beyond the Reparameterization Trick," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 2235–2244.

[93] D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[94] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[95] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258.

[96] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[97] K. Osawa, S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota, "Practical Deep Learning with Bayesian Principles," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[98] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[99] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks," *arXiv preprint arXiv:1602.02282*, vol. 3, no. 2, 2016.

[100] F. Wenzel, K. Roth, and B. S. Veeling, "How Good is the Bayes Posterior in Deep Neural Networks Really?" pp. 10 248–10 259, 2020.

[101] M.-H. Laves, M. Tölle, A. Schlaefer, and S. Engelhardt, "Posterior Temperature Optimization in Variational Inference for Inverse Problems," *Workshop in Advances in Approximate Bayesian Inference*, 2021.

[102] L. Noci, K. Roth, G. Bachmann, S. Nowozin, and T. Hofmann, "Disentangling the Roles of Curation, Data-Augmentation and the Prior in the Cold Posterior Effect," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 12 738–12 748, 2021.

[103] R. Kurle, R. Herbrich, T. Januschowski, Y. B. Wang, and J. Gasthaus, "On the Detrimental Effect of Invariances in the Likelihood for Variational Inference," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 4531–4542, 2022.

[104] B. Coker, W. P. Bruinsma, D. R. Burt, W. Pan, and F. Doshi-Velez, "Wide Mean-Field Bayesian Neural Networks Ignore the Data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2022, pp. 5276–5333.

[105] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[106] D. Krueger, C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville, "Bayesian Hypernetworks," *Workshop in Advances in Approximate Bayesian Inference*, 2017.

[107] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing Flows for Probabilistic Modeling and Inference," *Journal of Machine Learning Research (JMLR)*, vol. 22, no. 57, pp. 1–64, 2021.

[108] D. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 1530–1538.

[109] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-Linear Independent Components Estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[110] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved Variational Inference with Inverse Autoregressive Flow," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.

[111] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[112] C. Louizos and M. Welling, "Multiplicative Normalizing Flows for Variational Bayesian Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 2218–2227.

[113] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson, "Subspace Inference for Bayesian Deep Learning," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR, 2020, pp. 1169–1179.

[114] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[115] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[116] J. Watson, J. A. Lin, P. Klink, J. Pajarinen, and J. Peters, "Latent Derivative Bayesian Last Layer Networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2021, pp. 1198–1206.

[117] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable Bayesian Optimization Using Deep Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 2171–2180.

[118] M. Lázaro-Gredilla and A. R. Figueiras-Vidal, "Marginalized Neural Network Mixtures for Large-Scale Regression," *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1345–1351, 2010.

[119] N. Weber, J. Starc, A. Mittal, R. Blanco, and L. Màrquez, "Optimizing Over a Bayesian Last Layer," in *Workshop in Advances in Approximate Bayesian Inference*, 2018.

[120] E. Daxberger, E. Nalisnick, J. U. Allingham, J. Antorán, and J. M. Hernández-Lobato, "Bayesian Deep Learning via Subnetwork Inference," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 2510–2521.

[121] M. Sharma, S. Farquhar, E. Nalisnick, and T. Rainforth, "Do Bayesian Neural Networks Need to be Fully Stochastic?" in *Proceedings of the International*

Conference on Artificial Intelligence and Statistics (AISTATS). PMLR, 2023, pp. 7694–7722.

[122] M. F. Pradier, W. Pan, J. Yao, S. Ghosh, and F. Doshi-Velez, "Projected BNNs: Avoiding Weight-Space Pathologies by Learning Latent Representations of Neural Network Weights," *arXiv preprint arXiv:1811.07006*, 2018.

[123] M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran, "Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 2782–2792.

[124] Y. Wen, D. Tran, and J. Ba, "Batchensemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[125] Z. Deng and J. Zhu, "Bayesadapter: Being Bayesian, Inexpensively and Reliably, via Bayesian Fine-Tuning," in *Asian Conference on Machine Learning (ACML)*. PMLR, 2023, pp. 280–295.

[126] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational Convolutional Neural Network Pruning," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2780–2789.

[127] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[128] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end Object Detection with Transformers," in *Proceedings of the IEEE European Conference in Computer Vision (ECCV)*, 2020, pp. 213–229.

[129] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[130] X. Wang, S. Zhang, Z. Yu, L. Feng, and W. Zhang, "Scale-Equalizing Pyramid Convolution for Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 359–13 368.

[131] T. Kong, F. Sun, C. Tan, H. Liu, and W. Huang, "Deep Feature Pyramid Reconfiguration for Object Detection," in *Proceedings of the IEEE European Conference in Computer Vision (ECCV)*, 2018, pp. 169–185.

[132] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang, "Dynamic Head: Unifying Object Detection Heads with Attentions," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7373–7382.

[133] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng, and J. Sun, "You Only Look One-Level Feature," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 039–13 048.

[134] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction Without Convolutions," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021, pp. 568–578.

[135] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8759–8768.

[136] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, "M2det: A Single-Shot Object Detector Based on Multi-Level Feature Pyramid Network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 9259–9266.

[137] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7036–7045.

[138] Z. Deng, Y. Dong, S. Zhang, and J. Zhu, "Understanding and Exploring the Network with Stochastic Architectures," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 14 903–14 914, 2020.

[139] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A Regularization Method for Convolutional Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[140] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of Neural Networks Using Dropconnect," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2013, pp. 1058–1066.

[141] J. Antorán, J. Allingham, and J. M. Hernández-Lobato, "Depth Uncertainty in Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 10 620–10 634, 2020.

[142] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable Architecture Search," *arXiv preprint arXiv:1806.09055*, 2018.

[143] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 1929–1958, 2014.

[144] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational Dropout Sparsifies Deep Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017, pp. 2498–2507.

[145] C. Louizos, K. Ullrich, and M. Welling, "Bayesian Compression for Deep Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[146] D. Miller, N. Sunderhauf, H. Zhang, D. Hall, and F. Dayoub, "Benchmarking Sampling-based Probabilistic Object Detectors," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, vol. 3, 2019, p. 6.

[147] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 1050–1059.

[148] R. Zhao, K. Wang, Y. Xiao, F. Gao, and Z. Gao, "Leveraging Monte Carlo Dropout for Uncertainty Quantification in Real-Time Object Detection of Autonomous Vehicles," *IEEE Access*, 2024.

[149] D. Miller, L. Nicholson, F. Dayoub, and N. Sünderhauf, "Dropout Sampling for Robust Object Detection in Open-Set Conditions," in *International Conference on Artificial Neural Networks (ICANN)*. IEEE, 2018, pp. 3243–3249.

[150] A. Harakeh, M. Smart, and S. L. Waslander, "Bayesod: A bayesian approach for uncertainty estimation in deep object detectors," in *International Conference on Artificial Neural Networks (ICANN)*. IEEE, 2020, pp. 87–93.

[151] Z. Lyu, N. Gutierrez, A. Rajguru, and W. J. Beksi, "Probabilistic Object Detection via Deep Ensembles," in *Proceedings of the IEEE European Conference in Computer Vision (ECCV)*. Springer, 2020, pp. 67–75.

[152] F. Kuppers, J. Kronenberger, A. Shantia, and A. Haselhoff, "Multivariate Confidence Calibration for Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 326–327.

[153] B. Pathiraja, M. Gunawardhana, and M. H. Khan, "Multiclass Confidence and Localization Calibration for Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 19 734–19 743.

[154] X. Du, X. Wang, G. Gozum, and Y. Li, "Unknown-Aware Object Detection: Learning What you Don't Know from Videos in the Wild," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13 678–13 688.

[155] S. Wang, J. Gao, B. Li, and W. Hu, "Narrowing the Gap: Improved Detector Training with Noisy Location Annotations," *IEEE Transactions on Image Processing*, vol. 31, pp. 6369–6380, 2022.

[156] M. E. Tipping, "The Relevance Vector Machine," in *Advances in Neural Information Processing Systems (NeurIPS)*, 1999, pp. 652–658.

[157] M. Titsias and M. Lázaro-Gredilla, "Doubly Stochastic Variational Bayes for Non-Conjugate Inference," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2014, pp. 1971–1979.

[158] K. B. Petersen, M. S. Pedersen *et al.*, "The Matrix Cookbook," *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008.

[159] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[160] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2014, pp. 740–755.

[161] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[162] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully Convolutional One-Stage Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 9627–9636.

[163] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open MMLab Detection Toolbox and Benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[164] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, Large Minibatch SGD: Training Imagenet in 1 Hour," *arXiv preprint arXiv:1706.02677*, 2017.

[165] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-Resolution Representations for Labeling Pixels and Regions," *arXiv preprint arXiv:1904.04514*, 2019.

[166] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance Estimation for Neural Network Pruning," in *Proceedings of the IEEE International*

Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11 264–11 272.

[167] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[168] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "Plantdoc: A Dataset for Visual Plant Disease Detection," 2019.

[169] A. Crawshaw, "Uno Cards Dataset," https://public.roboflow.com/object-detection/uno-cards, 2020.

[170] A. Kirillov, K. He, R. Girshick, and P. Dollár, "A Unified Architecture for Instance and Semantic Segmentation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[171] G. Sfikas, A. P. Giotis, G. Retsinas, and C. Nikou, "Quaternion generative adversarial networks for inscription detection in byzantine monuments," in *International Conference on Pattern Recognition Workshops*. Springer, 2021, pp. 171–184.

[172] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv preprint arXiv:1411.1784*, 2014.

[173] H. Ritter, A. Botev, and D. Barber, "Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[174] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on Bayesian Neural Networks: A Tutorial for Deep Learning Users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[175] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A Survey of Uncertainty in Deep Neural Networks," *Artificial Intelligence Review*, pp. 1–77, 2023.

[176] E. Dupont, Y. W. Teh, and A. Doucet, "Generative Models as Distributions of Functions," *arXiv preprint arXiv:2102.04776*, 2021.

[177] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NERF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Proceedings of the IEEE European Conference in Computer Vision (ECCV)*, pp. 405–421, 2020.

[178] D. W. Romero, R.-J. Bruintjes, J. M. Tomczak, E. J. Bekkers, M. Hoogendoorn, and J. C. van Gemert, "Flexconv: Continuous Kernel Convolutions with Differentiable Kernel Sizes," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[179] N. Benbarka, T. Höfer, and A. Zell, "Seeing Implicit Neural Representations as Fourier Series," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 2041–2050.

[180] Y. Strümpler, J. Postels, R. Yang, L. V. Gool, and F. Tombari, "Implicit Neural Representations for Image Compression," in *Proceedings of the IEEE European Conference in Computer Vision (ECCV)*. Springer, 2022, pp. 74–91.

[181] D. J. MacKay, "A Practical Bayesian Framework for Backpropagation Networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[182] T. Karaletsos, P. Dayan, and Z. Ghahramani, "Probabilistic Meta-Representations of Neural Networks," *arXiv preprint arXiv:1810.00555*, 2018.

[183] F. Huszár, "Variational Inference Using Implicit Distributions," *arXiv preprint arXiv:1702.08235*, 2017.

[184] J. Shi, S. Sun, and J. Zhu, "Kernel Implicit Variational Inference," *arXiv preprint arXiv:1705.10119*, 2017.

[185] T. Karaletsos and T. D. Bui, "Hierarchical Gaussian Process Priors for Bayesian Neural Network Weights," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 17 141–17 152, 2020.

[186] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn, "CKConv: Continuous Kernel Convolution for Sequential Data," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[187] E. Dupont, A. Goliński, M. Alizadeh, Y. W. Teh, and A. Doucet, "Coin: Compression with Implicit Neural Representations," *arXiv preprint arXiv:2103.03123*, 2021.

[188] Z. Guo, G. Flamich, J. He, Z. Chen, and J. M. Hernández-Lobato, "Compression with Bayesian Implicit Neural Representations," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[189] J. Shen, A. Ruiz, A. Agudo, and F. Moreno-Noguer, "Stochastic Neural Radiance Fields: Quantifying Uncertainty in Implicit 3D Representations," in *International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 972–981.

[190] F. Vasconcelos, B. He, N. M. Singh, and Y. W. Teh, "UncertaINR: Uncertainty Quantification of End-to-End Implicit Neural Representations for Computed Tomography," *Transactions on Machine Learning Research*, 2023.

[191] E. Dupont, H. Kim, S. A. Eslami, D. J. Rezende, and D. Rosenbaum, "From Data to Functa: Your Data Point is a Function and You Can Treat It Like One," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 5694–5725.

[192] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density Estimation Using Real NVP," *arXiv preprint arXiv:1605.08803*, 2016.

[193] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[194] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep Ensembles: A Loss Landscape Perspective," *arXiv preprint arXiv:1912.02757*, 2019.

[195] A. Y. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner, "'in-Between' Uncertainty in Bayesian Neural Networks," *arXiv preprint arXiv:1906.11537*, 2019.

[196] A. Asuncion and D. Newman, "UCI Machine Learning Repository," 2007.

[197] A. Krizhevsky, G. Hinton *et al.*, "Learning Multiple Layers of Features from Tiny Images," 2009.

[198] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[199] M. S. A. Nadeem, J.-D. Zucker, and B. Hanczar, "Accuracy-rejection curves (arcs) for comparing classification methods with a reject option," in *Workshop of the International Conference on Machine Learning (ICMLW)*. PMLR, 2009, pp. 65–81.

[200] A. Foong, D. Burt, Y. Li, and R. Turner, "On the Expressiveness of Approximate Inference in Bayesian Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 15 897–15 908, 2020.

[201] C. C. Margossian and L. K. Saul, "The Shrinkage-Delinkage Trade-Off: An Analysis of Factorized Gaussian Approximations for Variational Inference," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR, 2023, pp. 1358–1367.

[202] V. M.-H. Ong, D. J. Nott, and M. S. Smith, "Gaussian Variational Approximation With a Factor Covariance Structure," *Journal of Computational and Graphical Statistics*, vol. 27, no. 3, pp. 465–478, 2018.

[203] C. Louizos and M. Welling, "Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 1708–1716.

[204] S. Sun, C. Chen, and L. Carin, "Learning Structured Weight Uncertainty in Bayesian Neural Networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017, pp. 1283–1292.

[205] A. Garriga-Alonso and M. van der Wilk, "Correlated Weights in Infinite Limits of Deep Convolutional Neural Networks," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR, 2021, pp. 1998–2007.

[206] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernandez-Lobato, and A. L. Gaunt, "Deterministic Variational Inference for Robust Bayesian Neural Networks," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[207] S. Farquhar, M. A. Osborne, and Y. Gal, "Radial Bayesian Neural Networks: Beyond Discrete Support In Large-Scale Bayesian Deep Learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020, pp. 1352–1362.

[208] D. Romero, D. D. Ariananda, Z. Tian, and G. Leus, "Compressive Covariance Sensing: Structure-Based Compressive Sensing Beyond Sparsity," *IEEE Signal Processing Magazine*, vol. 33, no. 1, pp. 78–93, 2015.

[209] Y. C. Eldar, J. Li, C. Musco, and C. Musco, "Sample Efficient Toeplitz Covariance Estimation," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2020, pp. 378–397.

[210] J. P. Cunningham, K. V. Shenoy, and M. Sahani, "Fast Gaussian Process Methods for Point Process Intensity Estimation," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2008, pp. 192–199.

[211] Y. Saatçi, "Scalable Inference for Structured Gaussian Process Models," Ph.D. dissertation, University of Cambridge, 2012.

[212] W. Tebbutt, T. D. Bui, and R. E. Turner, "Circular Pseudo-Point Approximations for Scaling Gaussian Processes," in *Workshop in Advances in Approximate Bayesian Inference*, 2016, pp. 1–5.

[213] A. G. Wilson, C. Dann, and H. Nickisch, "Thoughts on Massively Scalable Gaussian Processes," *arXiv preprint arXiv:1511.01870*, 2015.

[214] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An Exploration of Parameter Redundancy in Deep Networks with Circulant Projections," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2857–2865.

[215] V. Sindhwani, T. Sainath, and S. Kumar, "Structured Transforms for Small-Footprint Deep Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[216] S. Liao and B. Yuan, "Circconv: A Structured Convolution with Low Complexity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 4287–4294.

[217] S. Liao, A. Samiee, C. Deng, Y. Bai, and B. Yuan, "Compressing Deep Neural Networks Using Toeplitz Matrix: Algorithm Design and FPGA Implementation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 1443–1447.

[218] A. Thomas, A. Gu, T. Dao, A. Rudra, and C. Ré, "Learning Compressed Transforms with Low Displacement Rank," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[219] L. Zhao, S. Liao, Y. Wang, Z. Li, J. Tang, and B. Yuan, "Theoretical Properties for Neural Networks with Weight Matrices of Low Displacement Rank," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 4082–4090.

[220] M. Kissel and K. Diepold, "Structured Matrices and Their Application in Neural Networks: A Survey," *New Generation Computing*, vol. 41, no. 3, pp. 697–722, 2023.

[221] J. Harrison, J. Willes, and J. Snoek, "Variational Bayesian Last Layers," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[222] J. Li, Z. Miao, Q. Qiu, and R. Zhang, "Training Bayesian Neural Networks with Sparse Subspace Variational Inference," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[223] G. E. Hinton and D. Van Camp, "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights," in *International Conference on Learning Theory (COLT)*, 1993, pp. 5–13.

[224] V. Fortuin, "Priors in Bayesian Deep Learning: A Review," *International Statistical Review*, 2022.

[225] J. McInerney, "An Empirical Bayes Approach to Optimizing Machine Learning Algorithms," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[226] M. Welling, C. Chemudugunta, and N. Sutter, "Deterministic Latent Variable Models and Their Pitfalls," in *International Conference on Data Mining (SDM)*. SIAM, 2008, pp. 196–207.

[227] A. Fang, S. Kornblith, and L. Schmidt, "Does progress on imagenet transfer to real-world datasets?" *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[228] S. D. Karthik, Maggie, "APTOS 2019 Blindness Detection," 2019. [Online]. Available: https://kaggle.com/competitions/aptos2019-blindness-detection

[229] Z. Anna, H. Brian, S. George, W. Jochen, E. Julia, C. Marc, K. Nicholas, C. Noel, C. Phil, and R. Veronica, "SIIM-ISIC Melanoma Classification," 2020. [Online]. Available: https://kaggle.com/competitions/siim-isic-melanoma-classification

[230] E. Mwebaze, J. Mostipak, Joyce, J. Elliott, and S. Dane, "Cassava Leaf Disease Classification," 2020. [Online]. Available: https://kaggle.com/competitions/cassava-leaf-disease-classification

[231] B.-H. Tran, S. Rossi, D. Milios, and M. Filippone, "All You Need is a Good Functional Prior for Bayesian Deep Learning," *Journal of Machine Learning Research (JMLR)*, vol. 23, pp. 1–56, 2022.

[232] X. Fan, S. Zhang, B. Chen, and M. Zhou, "Bayesian Attention Modules," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 16 362–16 376, 2020.

[233] M. Figurnov, S. Mohamed, and A. Mnih, "Implicit Reparameterization Gradients," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[234] R. Fathony, A. K. Sahu, D. Willmott, and J. Z. Kolter, "Multiplicative Filter Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[235] A. Kazemnejad, I. Padhi, K. Natesan Ramamurthy, P. Das, and S. Reddy, "The Impact of Positional Encoding on Length Generalization in Transformers," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.

[236] J. He, G. Flamich, Z. Guo, and J. M. Hernández-Lobato, "Recombiner: Robust and Enhanced Compression with Bayesian Implicit Neural Representations," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[237] Y. Shen, N. Daheim, B. Cong, P. Nickl, G. M. Marconi, C. Bazan, R. Yokota, I. Gurevych, D. Cremers, M. E. Khan *et al.*, "Variational Learning is Effective for Large Deep Networks," *arXiv preprint arXiv:2402.17641*, 2024.

[238] I. Bellido and E. Fiesler, "Do Backpropagation Trained Neural Networks have Normal Weight Distributions?" Springer, 1993, pp. 772–775.

[239] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving Adversarial Robustness via Promoting Ensemble Diversity," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 4970–4979.

[240] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez, "Quality of Uncertainty Quantification for Bayesian Neural Network Inference," *arXiv preprint arXiv:1906.09686*, 2019.

[241] G. W. Brier *et al.*, "Verification of Forecasts Expressed in Terms of Probability," *Monthly weather review*, vol. 78, no. 1, pp. 1–3, 1950.

[242] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining Well Calibrated Probabilities Using Bayesian Binning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

[243] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[244] R. El-Yaniv *et al.*, "On the Foundations of Noise-Free Selective Classification." *Journal of Machine Learning Research (JMLR)*, vol. 11, no. 5, 2010.

[245] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

Appendix A

Variational Feature Pyramid Networks

A.1 Using a Laplace prior

A.1.1 Laplace Distribution

Our first option for the weight-connection prior had to be a Gaussian distribution, due to its simplicity, good analytical properties, and implications of the central limit theorem [238]. Recent research however does point out to non-Gaussianity of neural network weights [11], and we have indeed conducted (preliminary) experiments towards this direction, especially focusing on heavy-tailed alternatives. We experimented with the Laplace distribution which has heavier tails than the Gaussian and is discontinuous at $w = \mu$. It is often used in the context of Lasso regression, where it encourages sparsity in the learned weights [28]. Assuming that the weights are i.i.d., we set the prior distributions to zero-mean Laplace:

$$p(W) = \prod_{i=1} p(w_i) \quad \text{where} \quad w_i \sim \text{Laplace}(0, 1). \tag{A.1}$$

The straightforward choice for the approximate variational distribution that satisfies the conditions for applying SGVB is the factorized Laplace distribution:

$$q(W) = \prod_{i=1} q(w_i) \quad \text{where} \quad w_i \sim \text{Laplace}(\mu_i, \beta_i). \tag{A.2}$$

Thus the set we wish to optimize are the variational parameters $\phi = \{\mu, \beta_i\}$. We can easily draw samples from the variational posterior:

$$w = \mu - \beta \text{sign}(\epsilon) \log(1 - 2|\epsilon| + \alpha) \quad \text{where} \quad \epsilon \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2}), \tag{A.3}$$

where the sign function evaluates the sign of ϵ and α is a parameter with small value introduce to provide numerical stability. The KL term of the VLB can be computed analytically as:

$$KL(q_\phi(W)||p(W)) = -\log(\beta) + |\mu| + \beta \exp(\frac{-|\mu|}{\beta}) - 1. \tag{A.4}$$

A.1.2 Experimental Results

Table A.1: Numerical results for object detection/segmentation trials on COCO [160]. Average precision and precision on different threshold and object sizes are shown, alongside with network size and inference time (measured in milliseconds), for proposed models and other feature pyramid variants.

Network	Model	AP	AP_{50}	AP_{70}	AP_S	AP_M	AP_L	Params	Inference
Faster RCNN	Laplace	0.312	0.517	0.328	0.179	0.346	0.392	1.602M	6.8 ± 0.10
Mask RCNN	Laplace	0.283	0.473	0.294	0.125	0.304	0.412	1.740M	6.9 ± 0.03

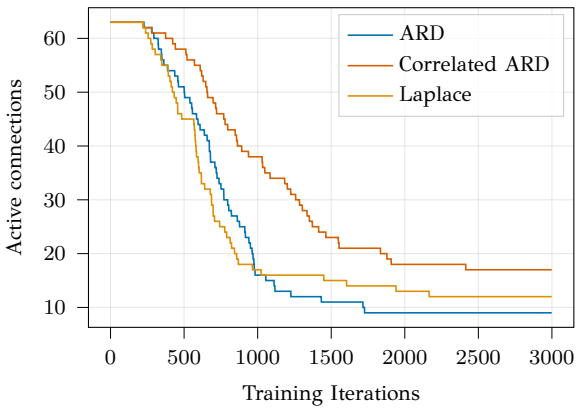

Figure A.1: Plot of the number of non-pruned weights/connections versus the training iterations using different priors on the same setting on COCO, (left Mask-RCNN and right Faster-RCNN).

Appendix B

Implicit Neural Representation Inference

B.1 INR Hypernetwork Details

This section delves deeper into the INR hypernetwork. We analyze its functionality and provide a graphical illustration of the process in Figure 4.2. Additionally, we present the training and inference procedures of our proposed low-dimensional inference scheme in two separate algorithms outlined in Figure B.1. .

As for the weight coordinates I, in practice these values are batched and computed separately for each layer. For the $i-th$ layer indices/input-coordinates positions have the shape $[n, I_{dims}]$ where n is the number of the total main network parameters of the $i-th$ layer and I_{dims} is the dimensionality of the indices. For example, for

Algorithm B.1 INR Training procedure

Require: I (Indices of main network weights), *Net* (Main network), *INR* (INR hypernetwork), *Dataset*.
for each Epoch **do**
  for (x, y) in *Dataset* **do**
    $y^{\star}$ = *Net*(x,ξ)
    loss = (y,$y^{\star}$)
    update *INR* w.r.t loss
    update *Net* w.r.t loss
  end for
end for

Algorithm B.2 INR Inference procedure

Require: I (Indices), *Net* (Main network), *INR* (INR hypernetwork), *Testset Approximate Inference* (Approximate inference method) *MC Samples* (Number of Monte Carlo samples).
for x in *Testset* **do**
  for j in range *MC Samples* **do**
    ξ_j = *Approximate Inference*(*INR*, I)
    $y^{\star}$ = *Net*(x,ξ_j)
  end for
end for
Calculate $y^{\star}$ statistics

Figure B.1: High level pseudo-code to introduce our method's behavior in training and inference settings (in this setting, a post-training Monte Carlo-based approximate inference method is implied).

a convolutional main layer $I_{dims} = 5$, the first 4 positions correspond to the kernel weights plus 1 dimension to act as the layer position (conditional position for each layer).

B.2 Experimental Setup

In this Section, we provide all the experimental details that were used in order to produce the results of the main paper. Our experimental setups and procedures are heavily influenced by current practices in the literature. In all experiments we chose the SIREN [2] network to serve as the INR hypernetwork, due its popularity and its ability to model highly complex signals without any use of positional encoding layers (for the coordinate tensors I). Furthermore, [2] described a hypernetwork initialization scheme for the w_{INR} parameters, that results in values ξ that are initially Normally distributed. This is in general beneficial for the training procedure and it is also common practice for initializing multiplicative noise [93, 147].

B.2.1 Design Choices

For each different modeling scenario we trained a ResNet-20 [20] on the CIFAR10 dataset. All models are trained using the Adam optimizer with learning rate equal to 10^{-3}, weight decay equal to 10^{-6} and a batch size equal to 256 running for 100 epochs. We evaluated the MAP solution for each model in clear and corrupted test data. As for evaluating the size of the INR hypernetwork we also deployed SWAG. We used a full Gaussian covariance to approximate the distribution of w_{INR}, and used 10 epochs of average with a learning step of 0.01. We evaluated the effect of the increasing size of hypernetworks by using the Log-likelihood, Expected Calibration Error and the Normalized Diversity. Concerning the latter, a typical way to quantify diversity is to compute the fraction of points where discrete predictions differ between two members, averaged over all possible pairs. This disagreement measure is normalized by $(1 - accuracy)$ of each prediction to take into account its sample predictive accuracy. Recent works [239, 194] point out that measuring the diversity of individual predictions obtained from each sampled network can highlight the quality of uncertainty.

B.2.2 Visualizing Uncertainty

This visualization task is highly suited to quantify "in-between" uncertainty of a model, as recent works found that standard numerical evaluation metrics such as log-likelihood struggle to fully capture this behavior, while at times overconfident methods may obtain better scores [240, 98]. We train a single, 2 hidden layer network, with 50 hidden ReLU units per layer using MAP inference until convergence. For the INR network $f_{w_{INR}}(\cdot)$ we used a SIREN [2] The INR network has 3 layers consisting of $[2, 10, 4]$ neurons respectively, resulting totally in 160 training parameters. Concerning the hyperparameters we used $\Omega_1 = 30$ for the first INR layer and $\Omega_l = 1$ for the rest while keeping the the parameter $c = 1$ fixed for all layers. The INR weights w_{INR} are initialized uniformly as $\sim \mathcal{U}(-\sqrt{c/n}/\Omega, \sqrt{c/n}/\Omega)$. The input coordinates $I \in \mathbb{R}^2$ are normalized to be in the range $[-1, 1]^2$. We optimize the Gaussian log-likelihood of our data, where the mean is produced by the network and the variance is a hyperparameter learnt jointly with NN parameters. We used a full batch Adam optimizer with a learning rate of 10^{-3}, $\alpha = 0.9$, $\beta = 0.999$ and weight decay $= 10^{-4}$. We trained all models for 600 epochs (since the amount of training samples are less than the actual training parameters). We used the same strategy for all of the baselines.

We deployed deep ensembles with an ensemble of 5 networks, as suggested by [193]. Dropout was set with dropping probability of 0.1. For the INR-RealNVP, following the literature, we tempered the posterior by applying a weight on the Kullback-Leibler term of the ELBO, equal to 0.1. For the INR equipped with the linearized Laplace we set the prior precision of $\lambda = 0.001$, where $C = \lambda^{-2} I_{d_\xi}$. Methods that required Monte Carlo sampling for estimating the predictive distribution use 30 MC samples during testing and 1 sample during training.

B.2.3 UCI Regression Benchmarks

We experiment with 8 UCI regression datasets using standard training-evaluation-test splits from [63] and their gap versions [195]. In this test, we use a fully-connected architecture with hidden layers that have $[50, 50, 20]$ neurons respectively followed by ReLU activation. All the training details are applied to all the regression datasets regardless of their individual characteristics such as size, input dimensions, etc. We used homoscedastic regression methods with a global variance parameter, $\mathcal{N}(y_i, g_w(x_i), \hat{\sigma}^2 I)$, where the logarithm of global log-variance $log\hat{\sigma}^2$ (in order to ensure positivity) is jointly optimized with the model parameters. Our training strategy follows [120]. We trained all methods for 50 epochs, except INR-RealNVP, which needed approximately 5 additional epochs to adapt. We employed early stopping if validation performance does not increase for 10 consecutive epochs. The weight settings which provide best validation performance in terms of log-likelihood are kept for testing. Again we used the Adam optimizer with a learning rate equal to 10^{-3}, weight decay equal to 10^{-6}, and a batch size equal to 512 samples. For the INR equipped with SWAG, we used full Gaussian covariance to approximate the distribution of w_{INR}, and used 25 epochs of average with a learning step of 0.01. For INR-Laplace, after trying several precision values we use a prior with a precision value $\lambda = 0.005$, as it yielded better validation results across all datasets.

B.2.4 Image Experiments

Through our image experiments we deployed the ResNet50 architecture [20]. As it is common practice, we applied several modifications to the original architecture such as replacing the kernel size of the first strided convolutional layer (7×7) to size 3×3. Additionally, we remove the first max-pooling layer. The rest of the ResNet details

were set according to [164]. For the INR network $f_{w_{INR}}(\cdot)$ we used a SIREN [2] shared across each layer of the main network. We used a variety of metrics, these include: test log-likelihood (LL); Brier score [241], which is a metric that measures accuracy of predictive probabilities by computing their mean squared distance from the one-hot class labels; the Expected Calibration Error (ECE, [242]), a metric which measures the difference between predictive confidence and empirical accuracy in classification. A detailed explanation of uncertainty evaluation metrics can be found in [141, 98, 193]. In our experiments we emphasized on out-of-distribution performance, as model that was well-calibrated on the training and validation distributions must ideally remain so on shifted data. Regarding the completely "out-of-distribution" (OOD) data, we expect the empirical entropy of the predicted distribution to be quite high. Essentially, a good model must be uncertain according to the degree that test inputs are far from the training distribution. For Dropout experiments, we add Dropout to the standard ResNet model in between the 2^{nd} and 3^{rd} convolutions in each ResNet block [98]. We used an ensemble of 5 elements for prediction. Ensemble elements differ from each other in their initialization, which is sampled from the He initialization distribution [243]. All models are trained using the Adam optimizer with learning rate equal to 10^{-3}, weight decay equal to 10^{-6}, with a batch size equal to 256 running for 50 epochs for MNIST and 150 epochs for the CIFAR10/CIFAR100 experiments respectively. The weight settings which provide best validation performance in terms of log-likelihood are kept for testing. During training, we also used plain data augmentation strategies including random image cropping and random horizontal flips. We used INR-SWAG for 10 epochs with learning rate equal to 10^{-4}. For INR-RealNVP, the base Gaussian distribution is set to $\mathcal{N}(0, 0.1I)$, transformed with a cascade of 4 coupling layers. Finally as for the experiments validating the uncertainty quality per low dimensional space we trained (each method) combined with a Resnet18 for 100 epochs in both Cifar10 and Cifar100 datasets while keeping the approximate inference method fixed same across all low dimensional spaces. While for each subspace method we followed the hyperparameters proposed in the original papers, for SWAG and Linearized Laplace with GGN, in order to be able to run across low dimensional spaces we choose the covariance to have Diagonal structure. We used SWAG for 10 epochs with learning rate equal to 10^{-3}. For the Laplace, we use a prior with a precision value $\lambda = 1.0$. All hyperparameters stayed the same across each method for comparison. Inference time (Table 4.2) for Resnet18 combined with different stochas-

tic subspaces and different approximate inference methods was measured in seconds and for a batch of 10 CIFAR images.

B.3 ReLU and Sinusoidal Hypernetworks

This section delves deeper into the activation function used in the hypernetworks. Our ablation study, focusing on SIREN activation, suggests that the hypernetworks need to model high-frequency representations of the weight perturbations. We begin by empirically quantifying the benefits of each activation type by evaluating the performance of the Maximum A Posteriori (MAP) estimate. We trained Resnet18 in both CIFAR10 and CIFAR100 for 100 epochs to evaluate the predictive capabilities of the Sinusoidal hypernetwork versus each ReLU counterpart.

Table B.1: Numerical results for classification trials with different hypernetwork activations.

Dataset	Hypernet Activation	Accuracy ↑	LL ↑	Error ↓	Brier ↓	ECE ↓
CIFAR10	ReLU	91.11	−0.48	0.08	0.14	0.05
	Sine	91.70	−0.44	0.08	0.13	0.05
CIFAR100	ReLU	67.79	−2.54	0.32	0.53	0.23
	Sine	68.49	−2.39	0.31	0.52	0.22

In Table B.1 we find that Sine/Periodic activations (the "default" choice in SIREN) slightly outperforms a hypernet with ReLU activations. Still, results are very close, though there is a trend in favor of sine in both benchmarks. The original motivation behind using the sine activation is related to modeling high-frequency content, which translates as details in structured signals such as images or video [2]. We can however see this "in the top of its head", so to speak: in structured signals we care more for low-frequency content, and high-frequency is a "good-to-have" content. We can interpret an input semantically if we see its low frequencies, but not necessarily vice versa. For example, image compression will invariably throw away high frequencies first, and the last frequencies to lose will be the lower ones. Our conjecture is as follows: When using an INR to model perturbations, we are faced with a different situation, that corresponds to a different "frequency landscape" (perhaps even different than the one of model weights). In particular, we do not have any reason to differentiate

lower or higher frequency content in any respect. We "care" for all frequencies, so we need to have a good way to model high frequencies as well. Perhaps this is the reason the sine activation gives a small edge over ReLU.

To elaborate further on this argument, we constructed a setting where we can visualize the ξ parameters and see if we can observe any meaningful connection between hypernetwork activation and frequencies modelled by the hypernetwork.

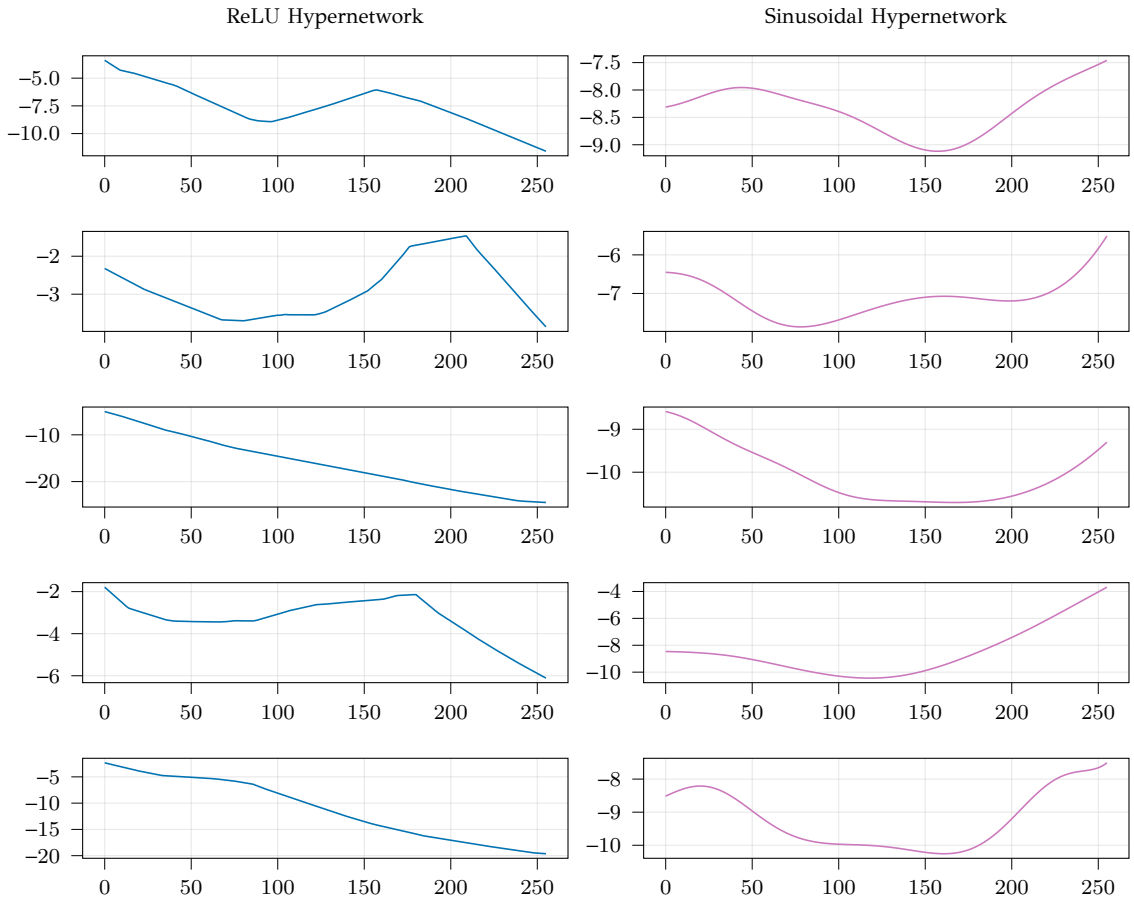

Figure B.2: Values of ξ as a function of input weight coordinates (channel-wise).

In Figure B.2 we plotted the the values of ξ as a function of input weight coordinates. Specifically for Resnet18 trained on CIFAR we plotted the flattened values for each specific kernel position across channels (channel slice) for 2 different convolutional layers. Both types of hypernetworks produce well structured perturbation functions. The ξ values produced from the sinusoidal hypernetwork are expressed as a somewhat oscillatory behavior w.r.t. channel position, which translates as higher frequency content. As for the ReLU perturbations, while having some high frequencies due to the discontinuity of the ReLU activation, the overall signal has a smooth

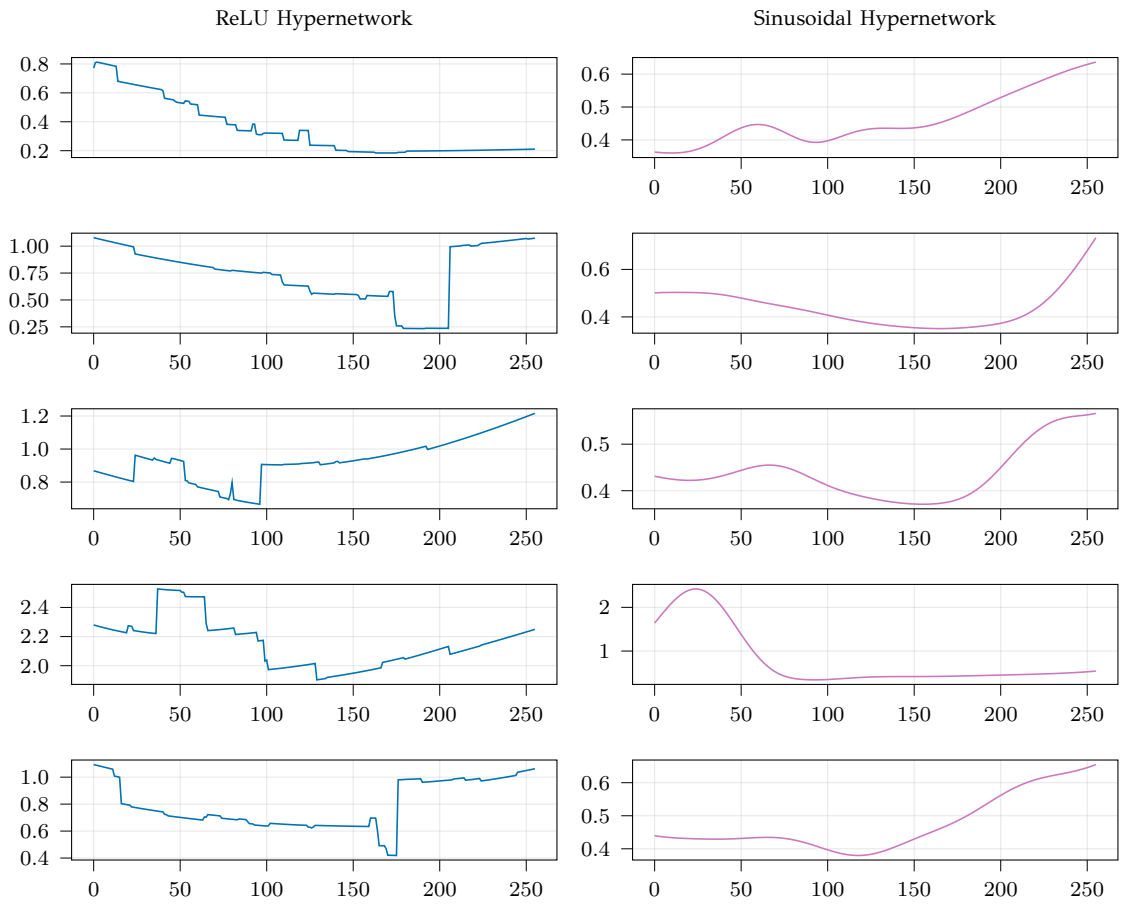

Figure B.3: Empirical variance of ξ as a function of input weight coordinates (channel-wise).

structure less complicated that the sinusoidal ones in some cases. Unsurprisingly, the ReLU result consists of practically piecewise linear components. This is what we believe that highlights the marginally better performance of SIREN hypernetworks. Furthermore, following the same experimental procedure we plotted alongside the mean values of ξ also their variance (Figure B.3), as this was computed from the SWAG-diagonal approximate inference method, again as a function of channel coordinates. We can observe that the variance has the same structural properties as the mean values of ξ. Thus, we believe that it makes sense for the main network convolutional kernel to take advantage of its structure.

B.4 Evaluating INR Hypernetwork Size

We added an ablation w.r.t. INR size following the UCI regression setting in our method 4.4. We compare four different versions of INR hypernetworks with an increasing number of parameters each, namely (BIG=2500, MED=625, SMALL=75, XSMALL=10), all combined with a Full GGN Laplace approximate posterior. From

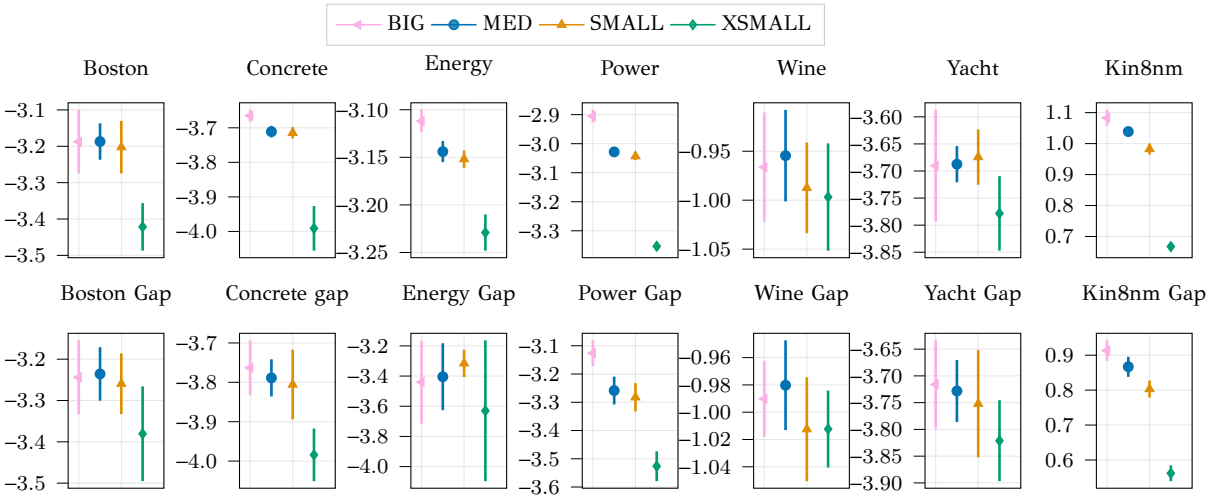

Figure B.4: Numerical results for regression trials on UCI standard [63] and GAP [195] splits for different hypernetwork sizes.

the experiments (Figure B.4) we can observe that there is a limit to where one can easily scale the INR hypernetwork and simultaneously gain performance. Individual characteristics play significant role to the INR size (main network size, dataset size, dataset dimension).

B.5 Computational Time

Regarding the computational time requirement of our method, it can be decomposed as follows:

$$t_{\text{Total}} = t_{\text{Hypernet evaluation}} + t_{\text{Approximate Inference}} \tag{B.1}$$

Where hypernetwork evaluation time according to Table B.2 makes the overall network in practice ≈ 1.2 slower than the vanilla network training. As for the approximate inference time although these methods we are using in our experiments are

expensive, because in our method they are applied in the small dimensional INR space in general it takes less time to evaluate. In Table B.3 we are considering the computational time of inference of our method versus standard inference popular ones.

Table B.2: Indicative time requirements for INR-based hypernetwork model.

Our method		Vanilla Network		Our method (fixed ξ perturbations)	
Forward	Backward	Forward	Backward	Forward	Backward
0.0069 ± 0.0001	0.014 ± 0.008	0.0046 ± 0.0001	0.011 ± 0.000	0.0045 ± 0.0002	0.009 ± 0.001

Table B.3: Computational time of INR low dimensional inference versus other approximate inference methods.

Method	Deep Ensembles	Dropout	LL Laplace	INR SWAG	INR RealNVP
Inference Time	0.9014 ± 0.0273	0.0372 ± 0.0066	2.0030 ± 0.0073	0.6393 ± 0.0184	0.2045 ± 0.0043

For the Deep Ensembles method the obtained values include additional overhead such as ensemble element loading etc. as it is common practice. Furthermore, the Linearized LL Laplace is much slower than the other methods as computing the Jacobian for the ResNet50 reaches the limits of our computational budget at this time.

As for the overhead in terms of learnable parameters, we have: W_{inr} (total number of the hypernetwork parameters), and q_{inr} (number of approximate inference parameters applied on the INR space), which as we mention in the main paper is in fact much less than q_W (number of approximate inference parameters applied on the full set of main network weights). Performance-wise our method is still being competitive w.r.t. methods like ensembles of D networks which at best is D times slower than the vanilla network.

Furthermore, because the main overhead of our method is the hypernetwork evaluation we investigated the following alternative training scheme, to further improve our method in terms of time. Instead of training the main network weights W and W_{INR} together we update the W_{INR} parameters every 10 epochs of the main network training, this significantly reduces the computational overhead of our method and we hypothesize it can scale to ImageNet models and datasets. Inference time for Resnet18 combined with different stochastic subspaces and different approximate

inference methods (time is measured in seconds and for a batch of 10 CIFAR images).

Table B.4: Numerical results for classification trials of ResNet18 in CIFAR100.

Training Scheme	Accuracy ↑	LL ↑	Error ↓	Brier ↓	ECE ↓
Full Training	69.01	−2.32	0.30	0.51	0.22
Alternative Training	68.59	−2.38	0.31	0.52	0.22

B.6 Additional Experiments

Further Image Experiments. Following [141, 120, 193], we train all methods on MNIST and CIFAR10 evaluate their predictive distributions on increasingly rotated digits. We trained the models for 50 epochs using the Adam optimizer. The results are depicted in Figure B.5. The importance of distributional shift expressed in this experiment via rotation of the original test set, which is highly informative as all methods perform more or less the same until the degradation shift reaches high intensity, where at this point methods begin to differentiate from one another. While the error of the prediction remains the same, metrics such as ECE and LL favor INR inference and Dropout which surpass the Deep Ensembles and LL Laplace as degradation increases significantly.

Table B.5: Numerical results for classification trials on different proposed low-dimensional spaces (CIFAR10).

Subspace	Inference	Standard				Corrupted			
		LL ↑	Error ↓	Brier ↓	ECE ↓	LL ↑	Error ↓	Brier ↓	ECE ↓
Rank1	SWAG	−0.41	0.08	0.13	0.05	−1.25	0.22	0.35	0.14
	Laplace	−1.56	0.09	0.68	0.68	−1.70	0.22	0.73	0.57
INR	SWAG	−0.32	0.07	0.12	0.04	−1.16	0.21	0.35	0.14
	Laplace	−1.56	0.11	0.68	0.66	−1.66	0.19	0.32	0.58
Subnetwork	SWAG	−0.42	0.07	0.12	0.04	−1.45	0.23	0.38	0.17
	Laplace	−1.55	0.09	0.68	0.68	−1.65	0.19	0.71	0.58
Partially stochastic	SWAG	−0.42	0.07	0.12	0.04	−1.44	0.20	0.38	0.17
	Laplace	−1.56	0.09	0.68	0.70	−1.67	0.21	0.72	0.59

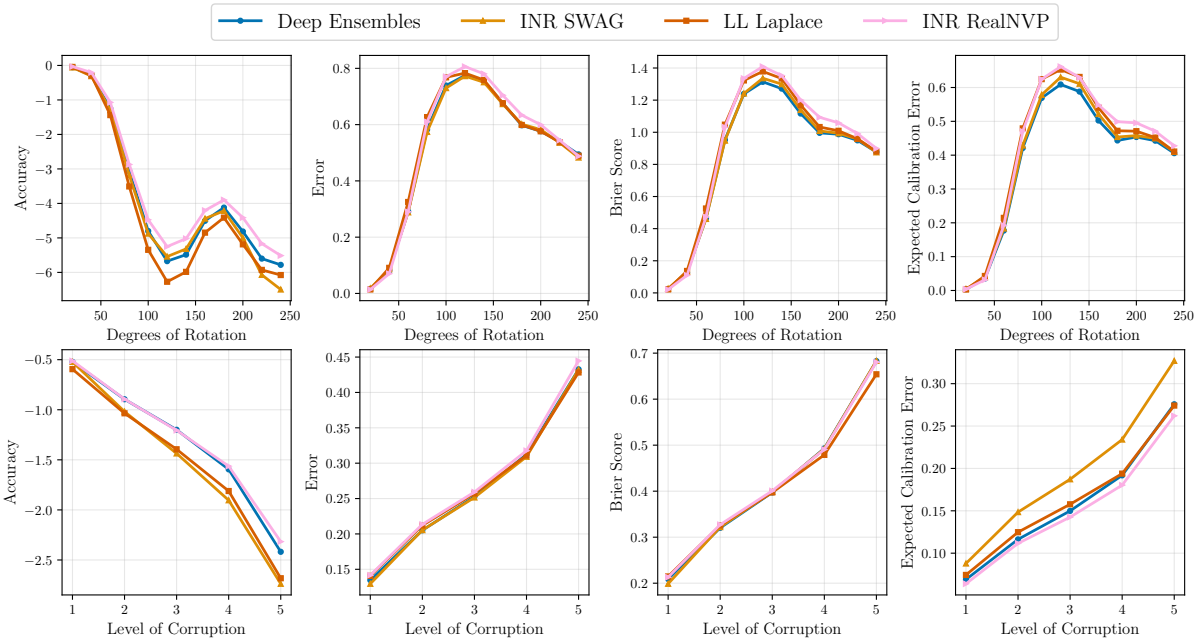

Figure B.5: Numerical results for classification trials on Rotated MNIST (top row) dataset and on Corrupted CIFAR10 (bottom row). Log-Likelihood (↑), Expected Calibration Error (↓), Brier Score (↓), Error (↓) and Accuracy (↑) are used for comparison. The x-axis of each plot corresponds to increasingly levels of corruption intensity.

B.7 Qualitative Evaluation of Empirical Densities

In this Section, we qualitatively inspect the approximate posterior distributions produced by INR variants in regression and classification settings. First, in Figure B.6 we plot the empirical density of $w \circ \xi$ for the network trained on the toy regression task. The variables are acquired by evaluating first eq:4.2 with 400 samples. Then we transform each sample according to $\xi = f_{INR}(\cdot)$ and finally scale the resulted values by w.

As we can see, INR-based models produce non-Gaussian approximate posterior distributions. Our results are in line with works like [224], which analyzed the empirical weight distributions of SGD-trained networks with different architectures, suggesting that fully connected neural networks learn heavy-tailed weight distributions.

We plotted empirical covariance matrices (see Figure B.9) that correspond to part of the $w \circ \xi$ parameters (specifically, the parameters that are "connected" to the first output neuron of the first layer of the main network). We can see that even the INR-based models are able to produce covariance matrices with high-magnitude

off-diagonal elements. This result validates the use of more expressive posterior distributions and highlights the performance of our hypernetwork method in the previous tasks.

We evaluated the empirical densities of convolutional layers following the classification setting of subsection 4.4.4. More specifically, we trained a ResNet-50 using the INR-RealNVP method on CIFAR10 dataset and evaluated the approximate distribution of $w \circ \xi$ for the first convolutional layer of the network, following the same sampling procedure as before. Results are depicted in Figure B.8, where the density histograms of the kernel values are Gaussian-like but still placing a lot of probability mass towards the tails.

We plotted the empirical covariance (Figure B.8 left) of values belonging to the same 3×3 kernel for nine different kernels. The covariance matrices indicated high spatial correlations of kernel values as was expected [224].

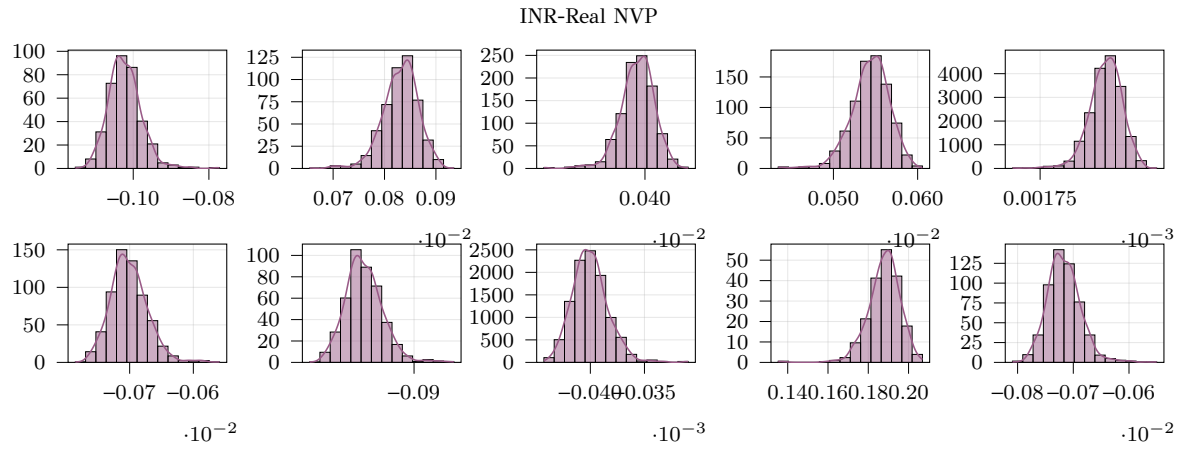

Figure B.6: Empirical Covariance for the INR-RealNVP for the first linear layer of the regression network.

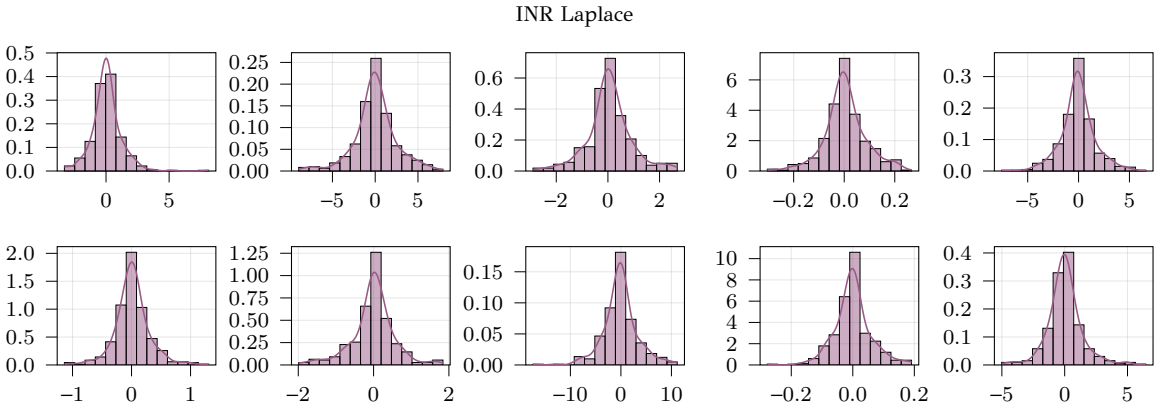

Figure B.7: Empirical Covariance for the INR-Laplace for the first linear layer of the regression network.

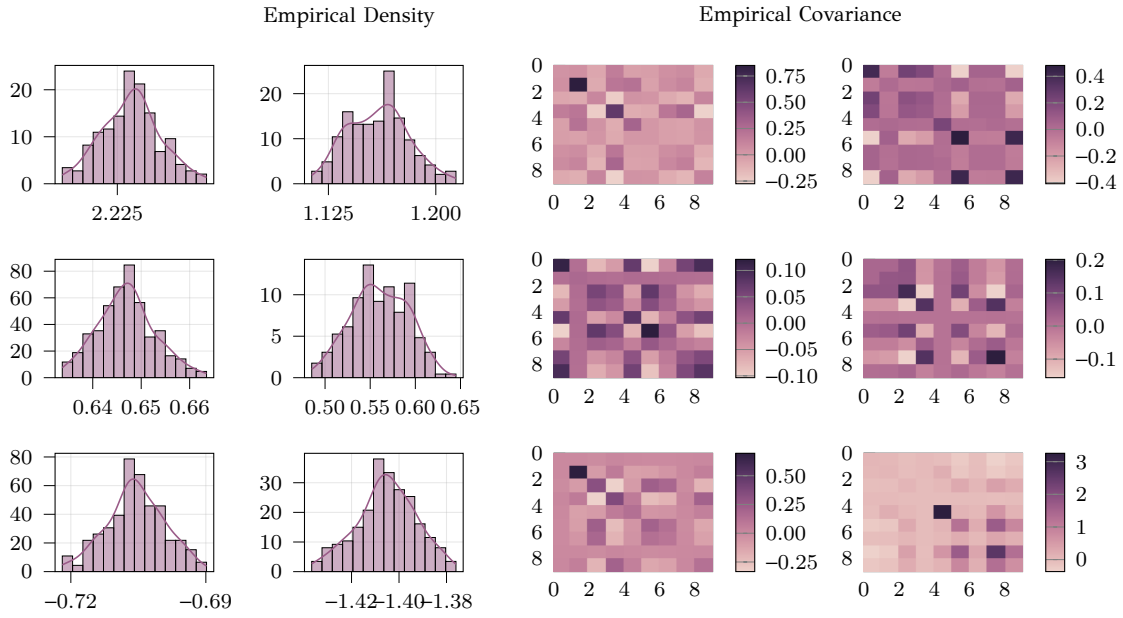

Figure B.8: Empirical density histogram and empirical covariance for of kernel values of the first convolutional layer of ResNet-50 using INR-RealNVP.

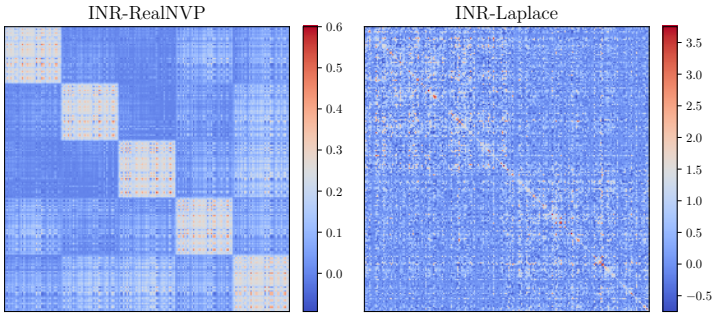

Figure B.9: Empirical Covariance for the INR-RealNVP and INR-Laplace for the first linear layer of the regression network.

Appendix C

Circulant Normal Approximate Distribution

C.1 Experimental Setup

In this Section, we provide all the experimental details that were used in order to produce the results of the main paper. Our experimental setups and procedures are heavily influenced by current practices in the literature. In all experiments we chose the standard variational inference. With tempered the kulbak-liebre term in Eq.5.2 by a factor λ, when $\lambda = 1$ the ELBO is a true lower bound to the marginal likelihood of the model, and the true posterior is recovered as the optimal solution when q is the family of all distributions over w. We use the pseudo-independent sampling method Flipout [96] for gradient variance reduction. In all experiments the hyperparameters for each method are optimized using the validation set. All the posteriors and prior are applied layer-wise thus we assume throughout the paper that the posterior and the prior over the neural network weights factorizes for each layer. We keep the weights with the produce the lowest error and use them for testing. Error bars and standard deviation number are acquired by averaging on three seeds We used a

variety of metrics, these include: test log-likelihood (LL); Brier score [241], which is a metric that measures accuracy of predictive probabilities by computing their mean squared distance from the one-hot class labels; the Expected Calibration Error [242], a metric which measures the difference between predictive confidence and empirical accuracy in classification. Also we included Selective prediction, a metric which modifies the standard prediction pipeline by introducing a "reject option" if the score for a prediction is below a certain threshold, the metric score is obtained by computing the area under the curve obtained for over a range of thresholds [244]

C.1.1 UCI Regression Benchmarks

We tested our method on 8 UCI regression datasets using established training-evaluation-test splits from [63] and their gap versions [195]. In this test, we use a fully-connected architecture with hidden layers that have [100, 50, 20] neurons respectively followed by ReLU activation. All the training details are applied to all the regression datasets regardless of their individual characteristics such as size, input dimensions, etc. We used homoscedastic regression methods with a global variance parameter, $N(y_i, g_w(x_i), \sigma^2 I)$, where the logarithm of global $\log \sigma^2$ variance log (in order to ensure positivity) is jointly optimized with the model parameters. We trained all methods for 100 epochs, we employed early stopping if validation performance does not increase for 10 consecutive epochs. The weight settings which provide best validation performance in terms of error are kept for testing. Again we used the Adam optimizer with a learning rate equal to $1e-3$, weight decay equal to 0.0001 momentum 0.9, and a batch size equal to 512 samples and we set the kl scale term $\lambda = 1$

C.1.2 Image Classification

Through our image experiments we deployed the ResNet50 architecture [20]. As it is common practice, we applied several modifications to the original architecture such as replacing the kernel size of the first strided convolutional layer (7×7) to size 3×3 and removing the first max-pooling layer. In our experiments we emphasized on out-of-distribution performance, as model that were well-calibrated on the training and validation distributions must ideally remain so on shifted data. Regarding the completely "out-of-distribution" (OOD) data, we expect the empirical entropy of the

predicted distribution to be quite high. Essentially, a good model must be uncertain according to the degree that test inputs are far from the training distribution. Following [193, 120, 141], we evaluate the models on both CIFAR10/CIFAR100 original test sets and also on the respective degraded sets. In the degraded sets, the original sets have been exposed to 16 different types of corruption with 5 levels of noise intensity each [198]. In the main paper for the corrupted version of the training dataset we used the concatenation of all 16 corrupted test sets with intensity 3. Random image cropping and horizontal flips are applied to each dataset during training alongside normalization using the training mean and standard deviation. All models are trained using the Adam optimizer with learning rate equal to $10-3$, weight decay equal to $10-6$, with a batch size equal to 64 running for 150 epochs with decreasing the learning rate by a factor 0.1 at epochs 80 and 120. The number Monte Carlo samples for estimating the predictive distribution was set to 10 while during training in order to estimate the reconstruction term (1st term of Eq 5.2) we use only one sample. The weight settings which provide best validation performance in terms of error are kept for testing.

For all the posterior methods we used fixed variance initialization for the variances while following the typical weight initialization for the mean values. Specifically for MFVI we set the posterior log variance -5.0 also for the K-Tied method. As for the ELRG we used a spherical diagonal with log variance -5.0 and low rank correlations were with 0.005 For the above methods we also used random initialization as well, but the where not a significant performance benefits. For the circulant covariance the initialization of kernel values affects in a great extend the predictive performance and the overall training stability. We used a fixed initialization scheme inspired from the Xavier method [245] where $k_i = 0.01 * \sqrt{2/d}$ where d is the over all number of weight in the layer. In all the experiments we used a circulant distribution with a kernel of $c = 20$ for the K-tied distribution we used $k = 3$ and for the ELRG we utilized rank-2 correlations. The kl anneal term λ through cross validation was set $\lambda = 0.001$ while for the ELRG method set to be $\lambda = 0.1$

For the transfer learning experiments we utilized the ImageNet pretrained weights for the ResNet 50. For each posterior distribution we initialized the means from the ImageNet weights and the covariance matrix for each method as in the previous experiments. Apart from the standard augmentations we specifically resized each image to 250×250 size. Following [227, 80] we used a class balanced random 20%

of the data as test data for every dataset. We fined tune each dataset for 30 epochs and report the prediction evaluate the model setting which provide the lowest error. For the experiments in Table 5.5 we adopted the rank-1 parameterization [123]. We used as a posterior distribution the log mixture following the paper. For the experiments which are considering additional priors first we used the prior from [11]. All convolutional weights of shape $In \times Out \times k \times k$ were set to follow the following Gaussian while the linear layers were left deterministic.

$$p(w) = N(0, \hat{\Sigma}) \quad \text{where} \quad \hat{\Sigma} = I \otimes B \tag{C.1}$$

$\sigma_{(i,j),(i,j')} = \exp(-||j - j'||_2))$ Where I has size of $in * out \times in * out$ and B has the size of k^2, k^2 This distribution is equivalent to a Matrix Normal prior on the convolutional weights. We used the same hyperparameters bu this time we set the $\lambda = 0.0001$ As a second prior distribution we choose the induced functional prior from [231]. We to draw preferable functions from an radial basis function (RBF) kernel with length scale and variance set to 1 and we minimized the Wasserstein distance between BNN induced functions and GP functions (activations) w.r.t the prior variances as proposed in the main paper. We adopted the publicly available implementation After the training we set the prior variance as the once found via the optimization, the prior means to zero and proceed to train the methods following the same training protocols as before while setting $\lambda = 0.001$ for all methods

C.2 Empirical Bayes for the Circulant Normal

Assuming a Gaussian posterior distribution $q(w) \sim \mathcal{N}(\mu_w, \Sigma_w)$ and a prior distribution $p(w) \sim \mathcal{N}(\hat{\mu}_w, \hat{\Sigma}_w)$ for $w \sim \mathbb{R}^N$, under the variational inference scheme we set to optimize the variational lower bound $\mathcal{L}(w)$. Optimization w.r.t. the parameters of the variational distribution can be done via stochastic optimization. In Empirical Bayes in an Expectation Minimization fashion we seek the parameters of $p(w)$ that directly optimize the ELBO:

$$\begin{aligned} \frac{\partial \mathcal{L}(w)}{\partial \hat{\Sigma}_w} = 0 \Rightarrow \frac{\partial}{\partial \hat{\Sigma}_w} \mathbb{E}_{q(w)}(p(y|g_w(x^\star))) + \frac{\partial}{\partial \hat{\Sigma}_w} \mathrm{KL}(q(w)||p(w)) = 0 \\ \Rightarrow \frac{\partial}{\partial \hat{\Sigma}_w} \mathrm{KL}(q(w)||p(w)) = 0 \end{aligned} \tag{C.2}$$

Above we assumed that the mean of the prior is se to zero we want to optimize w.r.t. $\hat{\Sigma}_w$. As the first term does not depend on the prior parameters the ELBO optimization resorts to find $\hat{\Sigma}_w$ the KL divergence between the prior and the posterior. The kl divergence between these two Gaussian distributions. is defined as:

$$\mathrm{KL}(q(w)||p(w)) = \underbrace{\log(|\hat{\Sigma}_w|)}_{(1)} - \log(|\Sigma_w|) - N + \underbrace{\mathrm{Tr}(\hat{\Sigma}_w^{-1}\Sigma_w)}_{(2)} + \underbrace{(\mu_w - \hat{\mu}_w)^T\hat{\Sigma}_w^{-1}(\mu_w - \hat{\mu}_w)}_{(3)} \tag{C.3}$$

If now we assume that the matrix $\hat{\Sigma}_w$ is circulant, then the main goal is to find the values of the kernel $\hat{k}$ that describes, generates the prior covariance matrix. The kernel has n elements in total. From the properties of circulant matrices we can expand $\hat{\Sigma}_w$ as follows:

$$\hat{\Sigma}_w = \hat{k}_0 I + \hat{k}_1 P + \hat{k}_2 P^2 + \cdots + \hat{k}_n P^n \tag{C.4}$$

The matrix is decomposed as linear combination of powers of a special permutation matrix P. This special permutation matrix is called circular shift matrix. Using the above equation its enables as to find the exact analytical solution for each optimization problem. $\partial\,\mathrm{KL}(q(w)||p(w))/\partial\hat{k}_i$ where $\hat{k}_i$ are the kernel values of $\hat{\Sigma}_w^{-1}$. We rewrite and differentiate each annotated term of Eq.C.3 w.r.t $\hat{k}_i$.

$$\begin{aligned}
(1) &= \log(|\hat{\Sigma}_w|) = -\log(|\hat{\Sigma}_w^{-1}|) \\
&\overset{\partial\hat{k}_i}{\Longrightarrow} -\frac{\partial \log(|\hat{\Sigma}_w^{-1}|)}{\partial\hat{k}_i} = -\,\mathrm{Tr}(P^n(\hat{k}_0 I + \hat{k}_1 P + \hat{k}_2 P^2 + \cdots + \hat{k}_n P^n)) \\
&= \mathrm{Tr}(P^n(\hat{\Sigma}_w)) = -N\hat{k}_i
\end{aligned} \tag{C.5}$$

In the final step P^n shifts the $\hat{\Sigma}_w$ moving the $\hat{k}_i$ elements in the main diagonal thus the trace can be easily computed.

$$\begin{aligned}
(2) &= \mathrm{Tr}(\hat{\Sigma}_w^{-1}\Sigma_w) = \mathrm{Tr}(\hat{k}_0 I\Sigma_w) + \mathrm{Tr}(\hat{k}_1 P\Sigma_w) + \cdots + \mathrm{Tr}(\hat{k}_n P^n\Sigma_w) \\
&= \hat{k}_0\,\mathrm{Tr}(I\Sigma_w) + \hat{k}_1\,\mathrm{Tr}(P\Sigma_w) + \cdots + \hat{k}_n\,\mathrm{Tr}(P^n\Sigma_w) \\
&\overset{\partial\hat{k}_i}{\Longrightarrow} \mathrm{Tr}(P^i\Sigma_w)
\end{aligned} \tag{C.6}$$

$$\begin{aligned}
(3) &= (\mu_w - \hat{\mu}_w)^T\hat{\Sigma}_w^{-1}(\mu_w - \hat{\mu}_w) \\
&= (\mu_w - \hat{\mu}_w)^T(\hat{k}_0 I + \hat{k}_1 P + \cdots + \hat{k}_n P^n)(\mu_w - \hat{\mu}_w) \\
&\overset{\partial\hat{k}_i}{\Longrightarrow} (\mu_w - \hat{\mu}_w)^T(P^i)(\mu_w - \hat{\mu}_w)
\end{aligned} \tag{C.7}$$

Using the above we can substitute $(1), (2)$ and (3) in Eq. C.3 and compute the optimal prior values

$$\frac{\partial\,\mathrm{KL}(q(w)||p(w))}{\partial\hat{k}_i} = 0 \Rightarrow n\hat{k}_i = (\mu_w - \hat{\mu}_w)^T(P^i)(\mu_w - \hat{\mu}_w) + \mathrm{Tr}(P^i\Sigma_w) \tag{C.8}$$

C.3 Additional Experiments

This section delves deeper into the visualizations presented in the main paper. It offers supplementary figures to clarify the findings. We start by plotting the difference in entropy values under the semantic shift scenario is illustrated in Figure C.1. We plotted the density histogram of predictive entropy estimates obtained from different methods on the SVHN dataset (out-distribution) and CIFAR100 test set (in-distribution). Ideally, the two histograms must be highly separated. We include further visualization for more layers of trained ResNet-20 with ELRG posterior for both Isotropic and empirical circulant prior Figure C.2. Finally, we conclude with histogram plots showcasing the values of the learned covariance matrix in a ResNet-20 equipped with MFVI posterior, trained on CIFAR100 under different priors (Figure C.4 and Figure C.3).

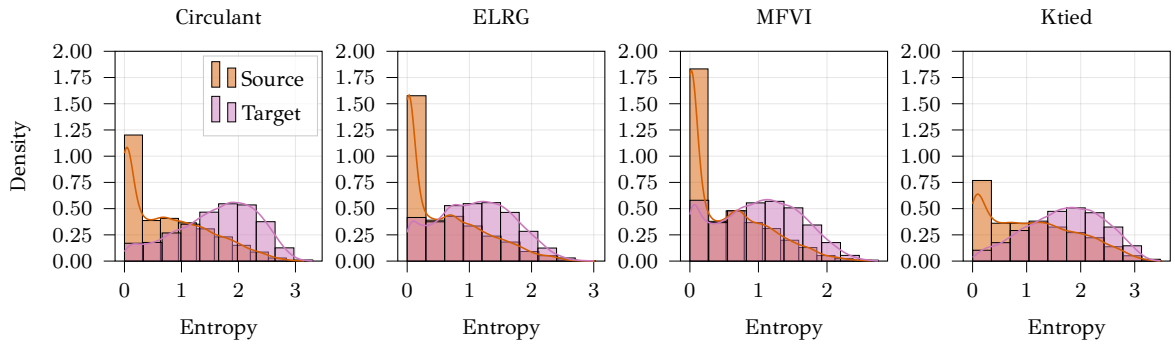

Figure C.1: Density histograms of predictive entropy estimates on CIFAR10 (in-distribution) and SVHN (out-distribution)

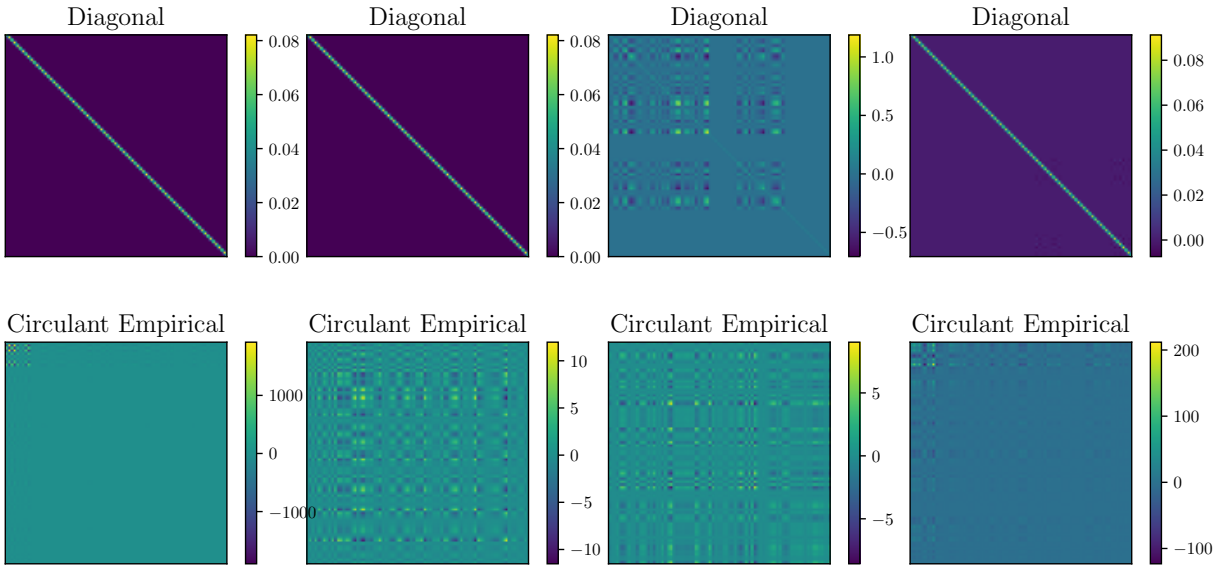

Figure C.2: Illustration of learned covariance matrices for the $1st$ convolutional layer of Resnet-20 trained on CIFAR100 for diagonal prior and for empirical circulant prior distribution.

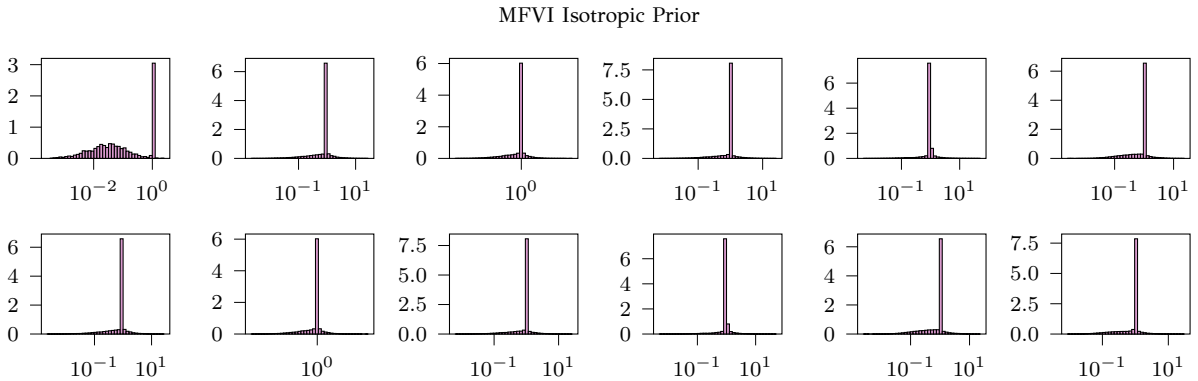

Figure C.3: Histogram plot of learned covariance matrix values in Resnet-20 equipped with MFVI posterior trained on CIFAR100 under isotropic prior.

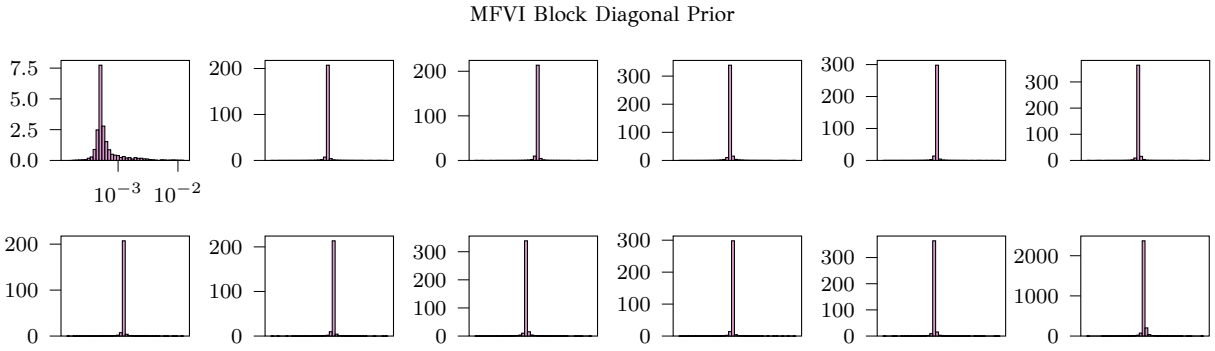

Figure C.4: Histogram plot of learned covariance matrix values in Resnet-20 equipped with MFVI posterior trained on CIFAR100 under the block diagonal prior of [11].

Author's Publications

1. P. Dimitrakopoulos, G. Sfikas, and C. Nikou. "**Implicit Neural Representation Inference for Low-Dimensional Bayesian Deep Learning.**". In Proceedings of the 12th International Conference on Learning Representations (ICLR), 2024.

2. G. Sfikas, G. Retsinas, P. Dimitrakopoulos, B. Gatos and C. Nikou, "**Shared-Operation Hypercomplex Networks for Handwritten Text Recognition.**", In Proceedings of the 17th International Conference on Document Analysis and Recognition (ICDAR), 2023, pp. 200-216.

3. P. Dimitrakopoulos, G. Sfikas, and C. Nikou. "**Variational feature pyramid networks**". In Proceedings of the 39th International Conference on Machine Learning (ICML), 2022, pp. 5142-5152.

4. P. Dimitrakopoulos, G. Sfikas, and C. Nikou. "**Wind: Wasserstein inception distance for evaluating generative adversarial network performance**", In Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 3182-3186.

5. P. Dimitrakopoulos, G. Sfikas, and C. Nikou. "**ISING-GAN: annotated data augmentation with a spatially constrained generative adversarial network**", In Proceedings of the 17th IEEE International Symposium on Biomedical Imaging (ISBI), 2020, pp. 1600-1603.

6. P. Dimitrakopoulos, G. Sfikas, and C. Nikou. "**Nuclei detection using residual attention feature pyramid networks**", In Proceedings of the 19th IEEE International Conference on Bioinformatics and Bioengineering (BIBE), 2019, pp. 109-114.

7. M. E Plissiti, P. Dimitrakopoulos, G. Sfikas, C. Nikou, O. Krikoni, and A. Charchanti. " **Sipakmed: A new dataset for feature and image based clas-**

sification of normal and pathological cervical cells in pap smear images", In Proceedings of the 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 3144-3148

Short Biography

Dimitrakopoulos Panagiotis received his M.Eng. and M.Sc. degrees in Computer Science from the Department of Computer Science and Engineering, University of Ioannina, Greece in 2019 and 2020, respectively. His M.Eng and M.Sc theses focused on Deep learning detection, classification models and Bayesian Variational models, both applied on the Medical Imaging domain. He is currently a Ph.D. candidate student in the same department from 2020. His theses is focused on the combination of Bayesian and deep learning methods for computer vision tasks. His research interests lie on Bayesian methods, Machine/Deep Learning, Computer Vision, Instance Segmantation/Detection, Medical Imaging.