

SISSA

Scuola
Internazionale
Superiore di
Studi Avanzati

Mathematics Area - PhD course in
Mathematical Analysis, Modelling, and
Applications

Generative Artificial Intelligence and Uncertainty Quantification for Scientific Discovery

Candidate:
Dario Coscia

Supervisor:
Prof. Gianluigi Rozza
Co-supervisors:
Prof. Max Welling
Nicola Demo

Academic Year 2025-26



Finanziato dall'Unione Europea – *Next Generation EU* - Missione 4, Componente 1 del Piano Nazionale di Ripresa e Resilienza (PNRR), Investimento 3.4 "Dottorati innovativi per le transizioni digitali e ambientali" - CUP G93C23000550003



ABSTRACT

Scientific machine learning models can achieve remarkable accuracy, yet fail silently when data are scarce, predictions extend beyond the training distribution, or errors accumulate during deployment. Together these limit their use in settings where an overconfident prediction could yield an unphysical result or misguide an expensive downstream experiment. This thesis investigates whether scientific machine learning models can be designed to recognise what they do not know, while remaining computationally efficient, scalable, and faithful to the physical structure of the underlying problem.

To address this question, this thesis develops efficient and scalable variational Bayesian methods across multiple modalities of scientific machine learning. These encompass causal dynamical systems, geometrically structured models subject to invariance and equivariance constraints, and large pretrained models adapted through parameter-efficient fine-tuning. Although these settings differ in their structure and computational requirements, they share the need to represent uncertainty without compromising the inductive biases that make scientific models effective.

The results demonstrate that principled Bayesian uncertainty need not require either prohibitively expensive posterior sampling or the loss of domain-specific structure. On the contrary, we show that it is possible to develop structured and computationally tractable Bayesian models for scientific machine learning. By treating uncertainty as a first-class component of scientific machine learning, this thesis lays the groundwork for a future generation of uncertainty-aware simulation, active learning pipelines, and scientific decision-making at scale.

CONTENTS

Abstract	iii
List of Figures	vii
List of Tables	xi
List of Acronyms	xv
1 Artificial Intelligence and Scientific Discovery	1
1.1 Scope & Research Questions	4
2 Background	9
2.1 A Probabilistic Approach to Deep Learning	9
2.2 Bayesian Inference in Deep Learning	13
3 Bayesian Autoregressive and Recurrent Neural Networks	23
3.1 Introduction	23
3.2 Background and Related Work	25
3.3 Methods	27
3.4 Experiments	33
3.5 Final Discussion	39
Appendix	43
3.6 Proofs and Derivations	43
3.7 BARNN for PDE Modelling Additional Results	48
3.8 BARNN for Molecule Generation Additional Results	54
3.9 Algorithms and Software	59
4 Bayesian Learned Interatomic Potentials	63
4.1 Introduction	63
4.2 Background and Related Work	65
4.3 Methods	67
4.4 Experiments	71
4.5 Final Discussion	77
Appendix	81
4.6 Proofs and Derivations	81
4.7 Dataset and Metrics	85

4.8	Architectures and Training Details	89
4.9	Algorithms and Software	91
4.10	Computational Cost Analysis	93
5	Bayesian Low-Rank Adaptation of Large Scale Models	97
5.1	Introduction	97
5.2	Background and Related Works	99
5.3	Methods	100
5.4	Experiments	105
5.5	Final Discussion	112
	Appendix	115
5.6	Proofs and Derivations	115
5.7	Experimental Details and Hyperparameters	120
5.8	Additional Results	126
6	Conclusions	129
6.1	Future Directions	130
6.2	Limitations	133
6.3	Closing Remarks	134
	List of Publications	135
	Disclosure on the Use of Large Language Models	137
	Bibliography	139

LIST OF FIGURES

- Figure 1 **Sources of scientific knowledge in ML.** *Ab-initio principles* embed physics directly; *Physical constraints* enforce governing equations or symmetries via loss constraints or equivariant architectures; *Correlations* capture spatiotemporal relationships in data; *Structural information* preserves topology in sequences or graphs. 3
- Figure 2 **Variational inference approximation.** Bayesian posterior estimation is recast as an optimisation problem by approximating the true posterior $p(\omega \mid \mathcal{D})$ with a parameterised distribution $q_{\phi}(\omega)$. The parameters ϕ are chosen to minimise their divergence from the true posterior. 16
- Figure 3 **BARNN generative and inference process.** Solid lines show the generative process (a), while dotted lines show the variational inference process (b). Network weights ω_t evolve jointly with the states y_t , forming a coupled probabilistic model. 24
- Figure 4 **Prediction and uncertainty intervals at the final time step.** CFD solution (red dotted line), BARNN mean prediction (solid blue line), and ± 3 standard-deviation confidence region (shaded blue) for each PDE. 35
- Figure 5 **RMSE and NLL across Neural PDE Solvers.** Despite comparable RMSE, NLL varies substantially across methods due to underestimated ensemble variance. 37
- Figure 6 **RMSE convergence with ensemble size in BARNN.** Accurate moment estimates are obtained with a small number of stochastic forward passes; a single MAP forward pass suffices when uncertainty is not required. 38

- Figure 7 **SMILES error analysis.** (a) Sources of invalid outputs and (b) validity as a function of ring count. Mean and standard deviation over four random-seed initialisations. 40
- Figure 8 **NLL and ECE convergence with ensemble size.** Both metrics stabilise with a small number of ensemble members, consistent with the RMSE trend in Figure 6. 53
- Figure 9 **Burgers rollout.** Representative one-dimensional solution over 320 autoregressive steps. 55
- Figure 10 **Kuramoto-Sivashinsky rollout.** Representative one-dimensional solution over 320 autoregressive steps. 55
- Figure 11 **Korteweg-de Vries rollout.** Representative one-dimensional solution over 320 autoregressive steps. 56
- Figure 12 **t-SNE of molecular descriptors.** BARNN and ChEMBL distributions overlap substantially in the latent space. 57
- Figure 13 **Per-descriptor property distributions.** KDE plots for BARNN and ChEMBL molecules show near-perfect overlap across all descriptors. 57
- Figure 14 **Validity vs. temperature.** BARNN maintains higher validity than baselines across most temperature values. 58
- Figure 15 **Uniqueness vs. temperature.** BARNN retains higher uniqueness at low to moderate temperatures. 58
- Figure 16 **Generated molecules.** Representative samples produced by BARNN. 59
- Figure 17 **Bayesian Learned Interatomic Potentials (BLIP).** BLIP introduces input-dependent stochastic perturbations to the weights of any message passing neural network, turning a deterministic MLIP into a principled probabilistic model trained via variational inference. 64
- Figure 18 **Correlation between error and uncertainty.** Spearman correlation between predicted uncertainty and squared force error in the OOD ammonia test set. 79

- Figure 19 **Forces MAE on silica across training-set sizes.** Fine-tuned ORB v3 Direct 2oOMAT results; error bars show the standard error over four random seeds. The best literature baseline is a four-member Deep Ensemble Equivariant PaiNN trained on 1024 configurations, with MAE 6.58 kcal/mol/Å. 80
- Figure 20 **Active-learning validation loss.** Validation loss across active-learning steps for three fine-tuning strategies, calculated as the sum of energy and force MAEs. Curves show means over three independent runs; energy is reported in eV and forces in eV/Å. 80
- Figure 21 **Overview of BaLoRA.** *Left:* The reduction matrix is treated as a random variable with input-adaptive Gaussian noise while the pretrained weights remain frozen. *Right:* The resulting uncertainty ellipsoid surrounds the LoRA estimate within the low-rank subspace, with its size determined by the input activation. 98
- Figure 22 **Prior-probability ablation.** Accuracy on LLaMA-3 across benchmark tasks for different values of the prior probability p in the KL-divergence loss. Results are robust across the tested values of p . 108
- Figure 23 **Bandgap prediction on QMOF.** Performance in eV for PEFT methods using MOFTransformer. 112
- Figure 24 **Inference-network encoder ablation.** Accuracy (top) and ECE (bottom) for four Vision Transformer encoders: ViT-Large, ViT-Medium, ViT-Small, and ViT-Tiny. 127
- Figure 25 **Monte Carlo convergence for bandgap prediction.** MAE and Spearman correlation as the number of test-time stochastic forward passes increases. No retraining is performed; shaded regions show ± 1 standard deviation across three random seeds. 128

LIST OF TABLES

Table 1	Prior ablation on sinusoidal time-series forecasting. RMSE, NLL, and ECE for different BARNN priors, Monte Carlo Dropout, and deterministic baselines. Static denotes a baseline in which the initial condition is used as the prediction at every time step. 34
Table 2	Uncertainty quantification for Neural PDE Solvers. NLL and ECE for Burgers, KS, and KdV over 320 autoregressive steps. Mean and std over four random seeds; <i>break</i> denotes divergence. 36
Table 3	De novo drug-like molecule generation. Validity, diversity, novelty, and uniqueness for BARNN and baselines. Mean and std over four random seeds. 39
Table 4	Molecular property distribution matching. Wasserstein distance W_2 between generated and training molecular property distributions. Mean and std over four random seeds. 39
Table 5	PDE discretisation parameters. n_t and n_x : time and space step sizes; $t_{\max}$: simulation duration; L : domain length. 49
Table 6	Dropout rate ablation. RMSE, NLL, and ECE on the validation set for 140-step unrolling across PDEs. 51
Table 7	Input perturbation scale ablation. RMSE, NLL, and ECE on the validation set for 140-step unrolling across PDEs. 51
Table 8	PDE Refiner hyperparameter ablation. RMSE, NLL, and ECE on the validation set for 140-step unrolling across PDEs. 52

Table 9	Pointwise accuracy across Neural PDE Solvers. RMSE for Burgers, KS, and KdV over 320 autoregressive steps. Mean and std over four random seeds. 54
Table 10	Results for the N-body system. Averages are computed over 4 random initialisation seeds. 72
Table 11	Energy and force MAE on OOD ammonia. Mean absolute errors for energy and forces, averaged over four random seeds. Baseline values are from (Tan et al. 2023). 74
Table 13	Energy and force MAE on MATPES PBE. Means and standard deviations over three random seeds. 76
Table 12	Calibration and error correlation on OOD ammonia. Expected calibration error and Spearman correlation for forces, averaged over four random seeds. Baseline values are from (Tan et al. 2023). 79
Table 14	Shared training hyperparameters. Main hyperparameters across experiments. Adam refers to the optimiser of Diederik P Kingma (2014), and AdamW to that of Loshchilov and Hutter (2017). 89
Table 15	Training and inference cost. Parameter counts and FLOPs for the ORBv3-inf-omat24 (Base) model, BLIP, and ensembles. 94
Table 16	Accuracy gains from Bayesian adaptation. Accuracy for standard LoRA and BaLoRA under equivalent dropout coefficients. Blue and red indicate percentage improvements and decreases relative to LoRA, respectively. 106
Table 17	Commonsense reasoning accuracy. Accuracy across PEFT methods and eight benchmarks. ChatGPT, LoRA, and DoRA results are from (Q. Huang et al. 2025); bold and underlined values indicate the best and second-best results, respectively. 107

Table 18	Image-classification accuracy. Accuracy across PEFT methods for ViT-L/16 on four datasets. Full fine-tuning is included as a reference; bold and underlined values indicate the best and second-best results, respectively. 109
Table 19	Image-classification calibration. Expected calibration error across PEFT methods for ViT-L/16 on four datasets. Bold and underlined values indicate the best and second-best results, respectively; values are percentages. 109
Table 20	Training resource comparison. Trainable parameters, peak memory, and FLOPs for PEFT methods on CIFAR-100 with ViT-L/16. Measurements use a single H100 94 GB GPU and one image input. 110
Table 21	Prior-probability ablation. Accuracy across common-sense reasoning benchmarks for different values of the prior probability p in the KL-divergence loss. 126

LIST OF ACRONYMS

AI Artificial Intelligence. 1, 3
ANN Artificial Neural Network. 9
BaLoRA Bayesian Low-Rank Adaptation. ix, xii, 97–101, 103–113, 121–128, 133
BARNN Bayesian Autoregressive and Recurrent Neural Network. vii, viii, xi, 23–25, 27, 29, 32–40, 47, 50, 52, 53, 57–59, 129, 131–133
BLIP Bayesian Learned Interatomic Potential. viii, 63–65, 67, 70–77, 79, 130, 131, 133
BMA Bayesian Model Averaging. 13, 20
BNN Bayesian Neural Network. 13, 18
DFT Density Functional Theory. 7
DL Deep Learning. 9, 13, 16
ELBO Evidence Lower Bound. 17
KL Kullback–Leibler. 11, 12, 17
LoRA Low-Rank Adaptation. ix, xii, 97–113, 115, 116, 118, 119, 121–125
ML Machine Learning. 1–3, 9, 10
MLE Maximum Likelihood Estimation. 11, 12
MLIP Machine Learning Interatomic Potential. 63–67
MPNN Message Passing Neural Network. 63, 65
NN Neural Network. 12, 13, 15, 20, 65
PDE Partial Differential Equation. 4, 5
SciML Scientific Machine Learning. 1, 2, 4–6, 14, 15, 25, 133, 134
SGD Stochastic Gradient Descent. 11
UQ Uncertainty Quantification. 3–6, 23, 25, 33, 47, 63, 66, 67
VAD Variational Adaptive Dropout. 68, 69, 90–93, 95, 100, 101

ARTIFICIAL INTELLIGENCE AND SCIENTIFIC DISCOVERY

The foundation of scientific discovery lies in the collection, transformation, and interpretation of data—an iterative process shaped by advances in methodology, technology, and social context. In recent decades, the emergence of Artificial Intelligence (AI) has introduced a paradigm shift, with AI increasingly becoming an integrated and, in some cases, indispensable tool in scientific research (H. Wang et al. 2023). In this light, Machine Learning (ML) plays a central role in modern scientific discovery by extracting patterns from large datasets, guiding experimentation, validating existing theories, generating data-consistent hypotheses, and automating key aspects of the discovery process.

Nevertheless, ML models purely *trained on* data often fail to respect underlying physical laws, while employing traditional physics-based solvers can be computationally expensive or limited in flexibility. The interdisciplinary field that bridges ML and physics-based solvers is known as Scientific Machine Learning (SciML) (Quarteroni et al. 2025), which embeds physical principles, constraints, correlations, and structural information directly into the learning framework. At present, SciML has already advanced a wide range of fields. In climate science (Bodnar et al. 2025; Price et al. 2025), large-scale atmospheric foundation models such as Aurora (Bodnar et al. 2025), pretrained on over a million hours of heterogeneous weather and climate data, have surpassed operational numerical weather prediction in global skill metrics while requiring only a fraction of the inference cost. In structural biology (Jumper et al. 2021; Abramson et al. 2024), ML systems have transformed the decades-old protein-

folding problem: AlphaFold (Jumper et al. 2021) achieves near-experimental accuracy in predicting three-dimensional protein structures from sequence alone, while its successor AlphaFold 3 (Abramson et al. 2024) extends this capability to jointly modelling proteins, nucleic acids, small molecules, ions, and modified residues, substantially outperforming previous specialised tools on protein–ligand and protein–nucleic acid interaction tasks and thereby accelerating both the mechanistic understanding of cellular processes and the rational design of therapeutics. In materials science (O. T. Unke et al. 2021; Neumann et al. 2024; Wood et al. 2025), machine-learned interatomic potentials deliver quantum-mechanical accuracy at a fraction of the cost of *ab initio* calculations: models such as Orb (Neumann et al. 2024), a family of universal neural network potentials that are 3–6 times faster than prior universal potentials, and UMA (Wood et al. 2025), trained on nearly half a billion unique atomic structures spanning molecules, materials, and catalysts, enable high-throughput geometry optimisation, molecular dynamics, and property prediction at scales previously inaccessible, significantly accelerating the computational discovery and design of new functional materials.

Despite these advances, SciML models face an undeniable challenge: even highly accurate ML models can produce *unreliable*, and in extreme cases *wrong*, predictions, with no mechanism to alert the practitioner. This phenomenon stems from several sources: *data scarcity*, whereby insufficient training data impedes learning, common in scientific contexts where data acquisition relies on costly experiments or simulations; *memorization*, whereby the model captures the physics of the training distribution yet fails to generalize to unseen scenarios; and *optimization*, whereby the model’s capacity is not fully exploited during training. Regardless of its origin, the model itself does not indicate that something has gone wrong. The out-of-distribution (OOD) problem is a particularly acute manifestation of these failure modes in SciML. Neural networks are known to assign high confidence to predictions even for inputs that lie far outside the training distribution (Tan et al. 2023), and in physical simulations, this pathology has direct consequences: ML interatomic potentials, for example, produce unphysical atomic configurations or exhibit biased dynamics precisely when evaluated in regions of configurational space that were never encountered during training. Because scientific applications routinely probe such extrapolative regimes – new materials compositions, unseen operating

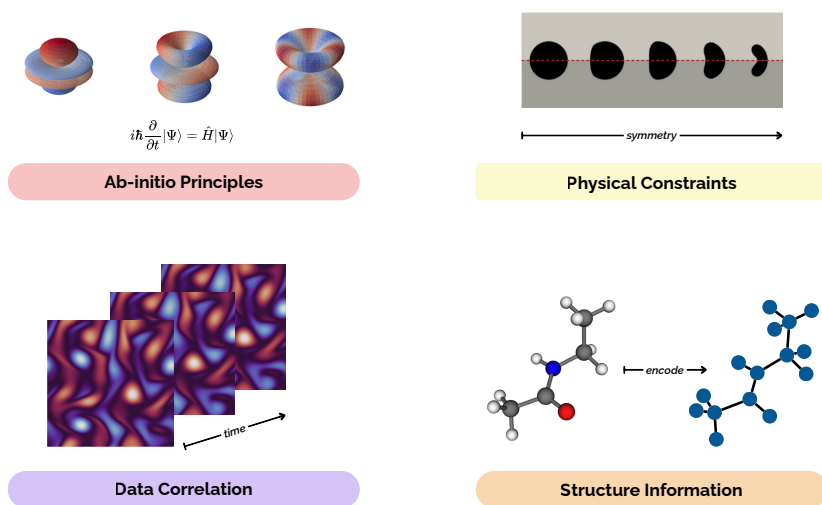


Figure 1: Sources of scientific knowledge in ML. *Ab-initio principles* embed physics directly; *Physical constraints* enforce governing equations or symmetries via loss constraints or equivariant architectures; *Correlations* capture spatiotemporal relationships in data; *Structural information* preserves topology in sequences or graphs.

conditions, or rare reaction pathways – the inability to detect OOD inputs without a principled uncertainty estimate is a fundamental barrier to reliable deployment. Understanding *when and how* a ML model makes mistakes is therefore crucial for detecting potential errors, such as unphysical solutions, when models silently violate conservation laws or boundary conditions, and yield completely unphysical results. Generative AI addresses this challenge naturally by modelling distributions over outputs rather than point estimates, providing a principled foundation for Uncertainty Quantification (UQ). Beyond diagnosing failures, this distributional perspective unlocks qualitatively new capabilities. In *inverse design*, rather than evaluating a fixed forward model to predict a response, one seeks to recover the input parameters, e.g., control variables, or boundary conditions, that produce a desired target output. Because such problems are typically ill-posed and admit multiple solutions, a probabilistic model is essential: it characterises the full distribution over

plausible designs and flags when proposed solutions venture outside the model’s reliable region (Pastrana et al. 2023; Schenck and Fox 2018). UQ also opens the door to *active learning*, whereby the model uses its own uncertainty to selectively query a classical solver or experimental oracle only for the most informative configurations, reducing the volume of costly labeled data needed to train or fine-tune it – an approach that has proven effective both for neural PDE solvers (Musekamp et al. 2025b) and for machine-learned interatomic potentials (Zaverkin et al. 2024). This probabilistic perspective finds its most rigorous expression in Bayesian modelling, which provides a coherent framework for incorporating prior knowledge, updating beliefs from data, and propagating uncertainty through predictions. Yet quantifying uncertainty in scientific applications remains an open and largely unsolved challenge. The function spaces involved are high-dimensional and physically constrained, making it difficult for models to remain expressive while respecting the underlying physics. Beyond expressiveness, computational cost poses a severe bottleneck: most existing Bayesian methods become intractable as problem size grows, limiting their applicability to small, idealised settings. Finally, even when uncertainty estimates can be obtained, they are often poorly calibrated, rendering them unreliable for the consequential decisions, such as experimental design or safety-critical predictions, that scientific applications demand.

1.1 SCOPE & RESEARCH QUESTIONS

In this thesis, we work towards developing computationally efficient and well-calibrated UQ methods, with a particular focus on scientific applications. As such, our main research question is:

Main Research Question: *Can we build SciML models that not only learn from data, but know what they do not know — producing calibrated uncertainty estimates, efficiently and at scale?*

Answering this question is harder than it may appear. Scientific machine learning models are deployed across a vast landscape of physical modalities, from time-dependent causal dynamical systems to geometrically symmetry-

based atomistic structures. Each domain imposes different constraints on what a model can represent, how it can be trained, and crucially, how uncertainty should be expressed and propagated. The challenge is not merely technical but foundational: in scientific applications, an overconfident prediction can silently violate conservation laws, produce unphysical solutions, or mislead a downstream decision such as an experimental campaign or a safety-critical simulation. At the same time, the scale and complexity of modern SciML models make naive Bayesian approaches computationally intractable, while simpler heuristics often fail to provide guarantees that practitioners can trust. There is no single UQ solution that fits all settings. We therefore decompose this central question into three research questions, each targeting a distinct class of SciML models:

Research Question 1: *How can we equip sequence-based models with efficient, calibrated uncertainty estimates while preserving their structure?*

Physical systems evolve causally: each state depends on what came before, and errors propagate forward with every autoregressive step. Consider a neural PDE solver predicting the evolution of a turbulent flow: at early time steps, the model may be accurate and confident, but as the solution drifts into uncharted regions of the state space, errors accumulate silently, with no mechanism to warn the practitioner. This silent failure is not merely inconvenient; in scientific applications, it can lead to qualitatively wrong conclusions drawn from seemingly plausible predictions. We address this in Chapter 3 by introducing BARNN, a variational Bayesian framework that transforms any autoregressive or recurrent model into its Bayesian counterpart. BARNN demonstrates that principled UQ over sequential models is both tractable and beneficial, producing calibrated, sharp uncertainty estimates at the cost of a single model, without retraining or architectural changes.

Yet temporal systems represent only one facet of scientific modelling. Many of the most challenging problems in computational science do not unfold sequentially, but are instead governed by complex, many-body physical interactions, where the relevant structure is not time, but space and geometry. This motivates our second research question:

Research Question 2: *How can we develop scalable Bayesian methods for SciML models defined on physically structured domains, such as atomistic systems, while remaining architecture-agnostic and computationally efficient?*

Consider the task of predicting how a material fractures under mechanical stress, or how a catalyst molecule binds to a reactive surface. These phenomena are governed by quantum mechanical interactions between atoms, interactions that are inherently geometric, many-body, and sensitive to symmetry. Machine learning interatomic potentials have emerged as powerful surrogates for such systems, but like most deep learning models, they fail silently: a potential trained on near-equilibrium configurations may confidently extrapolate to highly distorted structures it has never seen, producing physically implausible forces with no indication of error. This is particularly dangerous in active learning pipelines, where unreliable uncertainty estimates can steer data acquisition towards uninformative or even misleading configurations. In Chapter 4, we address this through BLIPs, an architecture-agnostic variational Bayesian framework for message passing neural networks, which injects input-adaptive Gaussian noise into the message and update functions while preserving physical symmetries. BLIPs demonstrates that principled UQ in atomistic systems is tractable, well-calibrated, and improves predictive accuracy, all at the inference cost of a standard deterministic model.

Nonetheless, both BARNN and BLIPs operate in a regime where models are trained from scratch on domain-specific data. The landscape of scientific machine learning is shifting: increasingly, large pretrained foundation models serve as the starting point, and task-specific adaptation is achieved through efficient fine-tuning. This raises our third question:

Research Question 3: *How can we endow large pretrained models with calibrated uncertainty estimates during fine-tuning, without sacrificing accuracy or scalability?*

Imagine a materials scientist who wishes to fine-tune a large pretrained model for predicting the electronic bandgap of metal-organic frameworks, a property critical for photocatalysis and energy storage. Full fine-tuning is computationally prohibitive, and so parameter-efficient methods such as Low-Rank Adaptation (LoRA) are the natural choice. Yet standard LoRA produces a single point estimate: it tells the scientist what the bandgap is predicted to be, but

not how much to trust that prediction. In a data-scarce scientific setting, where every mislabelled or uncertain prediction may cost days of DFT computation, this lack of calibrated uncertainty is a critical limitation. In Chapter 5 we address this through BaLoRA, a Bayesian extension of LoRA that introduces an input-adaptive reparametrization of the low-rank reduction matrix, treating its entries as random variables rather than fixed parameters. Remarkably, this Bayesian formulation not only yields well-calibrated, zero-shot uncertainty estimates at test time for scientific applications, but it also improves predictive accuracy, narrowing the gap with full fine-tuning across language and vision.

BACKGROUND

In this thesis, we aim to enable machine learning models the ability to know when they don't know. To achieve this, we train models using Bayesian learning. This chapter introduces the theoretical foundations of our work, including probabilistic modeling, neural network parameterizations of conditional distributions, and Bayesian inference. It also discusses current challenges in scientific machine learning and outlines the main contributions of this thesis.

2.1 A PROBABILISTIC APPROACH TO DEEP LEARNING

Over the past two decades, the emergence of Deep Learning (DL) has driven remarkable progress in the broader field of Machine Learning, transforming domains such as computer vision (Krizhevsky et al. 2012; He et al. 2016; Redmon et al. 2016; Dosovitskiy 2020), speech recognition (G. Hinton et al. 2012; Baevski et al. 2020), and natural language processing (Bengio, Ducharme, Vincent, and Jauvin 2003; Vaswani et al. 2017; Radford et al. 2018; Brown et al. 2020; Touvron, Lavril, et al. 2023). More recently, advancements in Artificial Neural Networks (ANNs) architectures have begun to reshape fields such as robotics, wireless communication, and the natural sciences (Scarselli et al. 2008; Simonyan and Zisserman 2014; Kipf and Welling 2017; Vaswani et al. 2017; Gilmer et al. 2017; Jaegle et al. 2021; M. Beck et al. 2024).

At its core, Machine Learning can be viewed through a probabilistic lens. In this perspective, the goal is to learn a model of the underlying probability distribution governing the data. In this setting, we assume the observed state $y \in \mathbb{Y}$ is a random sample from an unknown underlying process, whose true

distribution $p_{\text{data}}(\mathbf{y})$ is unknown. The ultimate goal of ML is to discover a model $p_{\theta}(\mathbf{y})$, with parameters $\theta \in \Theta$, that resembles the true unknown distribution. *Learning* is, most commonly, the process of adapting the parameters θ such that the true and model distribution match:

$$p_{\theta}(\mathbf{y}) \approx p_{\text{data}}(\mathbf{y}) \quad \forall \mathbf{y} \text{ observed.} \quad (1)$$

In particular, we aim to learn a model $p_{\theta}(\mathbf{y})$ that is expressive enough to generalize to unseen states, while also enabling the incorporation of inductive biases. For instance, in molecular modeling, we may enforce equivariance to rotations and translations so that predictions are consistent regardless of the molecule’s orientation.

Finally, in many cases, we are not interested in learning the distribution over the state $\mathbf{y}$, but rather how the state depends on some (random) variable $\mathbf{x}$. In other words, we seek to represent the probability of observing an output $\mathbf{y} \in \mathbb{Y}$ given an input $\mathbf{x} \in \mathbb{X}$, namely the *conditional distribution* $p_{\text{data}}(\mathbf{y} \mid \mathbf{x})$. A classic example is the *classification problem*, where the goal is to predict the label $\mathbf{y}$ given some input $\mathbf{x}$, for instance, predicting the class of an image, or determining whether a specific molecule is active or inactive with respect to a particular biological target.

For simplicity of notation, we will primarily focus on unconditional models. Nonetheless, the techniques presented in this work are applicable to conditional settings as well. In such cases, the conditioning data can be treated as additional inputs to the model, much like parameters, with the key difference that their values are fixed and not subject to optimization.

2.1.1 Dataset

We define a dataset $\mathcal{D}$ as a collection of $N \geq 1$ datapoints obtained by sampling from the true distribution $p_{\text{data}}(\mathbf{y})$:

$$\mathcal{D} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\} := \{\mathbf{y}_i\}_{i=1}^N. \quad (2)$$

When learning the conditional distribution $p_{\text{data}}(\mathbf{y} \mid \mathbf{x})$, the dataset is:

$$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N)\} := \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N. \quad (3)$$

Throughout this work, we assume that the dataset consists of independent and identically distributed (i.i.d.) random samples. In other words, the data consists of independent measurements of the same underlying physical phenomenon. Under this assumption, the probability of the data given the model parameters factorizes as a product of the probabilities of the individual datapoints.

2.1.2 Optimization

A common approach to training probabilistic models is Maximum Likelihood Estimation (MLE) (Bishop and Nasrabadi 2006). MLE aims to find the parameters θ that maximize the likelihood of the observed data:

$$\log p_{\theta}(\mathcal{D}) = \sum_{\mathbf{y} \in \mathcal{D}} \log p_{\theta}(\mathbf{y}). \quad (4)$$

This optimization is typically performed using gradient-based methods, where the parameters are iteratively updated according to the gradient of the log-likelihood:

$$\theta_{n+1} = \theta_n + \eta \nabla_{\theta} \log p_{\theta}(\mathcal{D}), \quad (5)$$

with $\eta > 0$ denoting the learning rate and $n \geq 0$ indexing the iterations. A more efficient approach is Stochastic Gradient Descent (SGD) and its variants (Robbins and Monro 1951; Bottou 2012), where the gradient is estimated using a small randomly sampled subset (mini-batch) of the data rather than the full dataset. This reduces computational cost per iteration and often improves convergence in practice. Specifically, for a mini-batch $\mathcal{B} \subset \mathcal{D}$, the parameter update becomes

$$\theta_{n+1} = \theta_n + \eta \nabla_{\theta} \sum_{\mathbf{y} \in \mathcal{B}} \log p_{\theta}(\mathbf{y}), \quad (6)$$

enabling scalable training on large datasets.

Finally, it is worth noting that estimating the maximum likelihood is equivalent to minimizing the Kullback–Leibler (KL) between the empirical data distribution $p_{\text{data}}(\mathbf{y})$ and the model $p_{\theta}(\mathbf{y})$. The KL divergence is a measure of how one probability distribution differs from another and is defined as

$$D_{\text{KL}}(p_{\text{data}} \parallel p_{\theta}) = \mathbb{E}_{\mathbf{y} \sim p_{\text{data}}} \left[\log p_{\text{data}}(\mathbf{y}) - \log p_{\theta}(\mathbf{y}) \right]. \quad (7)$$

Minimizing the KL divergence with respect to θ is equivalent to maximizing the log-likelihood of the observed data, since the term $\mathbb{E}_{\mathbf{y} \sim p_{\text{data}}} [\log p_{\text{data}}(\mathbf{y})]$ does not depend on θ . This connection is important because it provides a probabilistic interpretation of maximum likelihood: by optimizing the likelihood, we are effectively making the model distribution p_θ as close as possible to the true data distribution p_{data} , measured in terms of information loss.

2.1.3 Parametrizing Distributions with Neural Networks

Neural Networks (NNs) can be naturally used to parametrize probability distributions (Goodfellow et al. 2016), providing a flexible and expressive way to model complex data thanks to the universal approximation theorem (Hornik et al. 1989). In this approach, the NNs are part of the function that computes the probability density. As a concrete example, consider modeling a conditional Gaussian distribution for regression tasks. Suppose we want to model the conditional distribution of a continuous output $\mathbf{y} \in \mathbb{R}^d \subset \mathbb{Y}$ given an input $\mathbf{x} \in \mathbb{R}^m \subset \mathbb{X}$ as

$$\mathbf{y} \mid \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}), \text{diag}(\sigma^2)), \quad (8)$$

where σ^2 is the variance and $\boldsymbol{\mu}_\theta(\mathbf{x})$ is the mean function predicted by a NN with parameters θ . The likelihood of a single observation $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}$ is then

$$p_\theta(\mathbf{y}_i \mid \mathbf{x}_i) = \frac{1}{\sqrt{(2\pi)^d \prod_{j=1}^d \sigma_j^2}} \exp \left(-\frac{1}{2} \sum_{j=1}^d \frac{(y_{i,j} - \mu_{\theta,j}(\mathbf{x}_i))^2}{\sigma_j^2} \right). \quad (9)$$

Optimizing by Maximum Likelihood Estimation is equivalent to minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = -\sum_{i=1}^N \log p_\theta(\mathbf{y}_i \mid \mathbf{x}_i) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^d \frac{(y_{i,j} - \mu_{\theta,j}(\mathbf{x}_i))^2}{\sigma_j^2} + \text{const.} \quad (10)$$

Notice that if the variance σ^2 is constant, minimizing the negative log-likelihood is equivalent to minimizing the sum of squared errors between the predicted mean and the observed targets. In other words, standard mean-squared error training commonly used in regression is a special case of maximum likelihood estimation under a fixed-variance Gaussian model.

This example illustrates how NNs can parameterize the mean of a probabilistic model while keeping the variance fixed, enabling flexible conditional density estimation. While we show a Gaussian distribution here for simplicity, the same principle applies to other distributions, discrete or continuous, allowing NNs to model highly expressive and structured probability distributions in a wide range of scientific and engineering applications.

2.2 BAYESIAN INFERENCE IN DEEP LEARNING

Bayesian models have a long-standing role in Deep Learning (G. E. Hinton and Van Camp 1993; Welling and Teh 2011; Graves 2011; Blundell et al. 2015; Gal and Ghahramani 2016), providing a principled framework to model uncertainty in NNs through probabilistic inference. Unlike standard NNs, which learn deterministic point estimates of parameters θ , Bayesian Neural Networks (BNNs) treat the network parameters as random variables ω . Specifically, we assume

$$\omega \sim p(\omega), \quad (11)$$

where $p(\omega)$ is a prior distribution reflecting our initial beliefs about the network parameters before observing any data.

After observing a dataset $\mathcal{D}$, we can update these beliefs using Bayes' theorem (Bayes 1763) to obtain the *posterior distribution* over parameters:

$$p(\omega \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \omega) p(\omega)}{\int p(\mathcal{D} \mid \omega) p(\omega) d\omega}, \quad (12)$$

where $p(\mathcal{D} \mid \omega)$ is the likelihood of the data given a particular set of parameters. Intuitively, this formula tells us how to combine our prior knowledge about the parameters with the evidence provided by the data to obtain updated beliefs.

Once the posterior distribution is obtained, predictions for a new input $y_* \notin \mathcal{D}$ are made by Bayesian Model Averaging (BMA):

$$f^*(y_*) = \mathbb{E}_{p(\omega \mid \mathcal{D})} [f_\omega(y_*)] = \int f_\omega(y_*) p(\omega \mid \mathcal{D}) d\omega, \quad (13)$$

for any parametrized model $f_\omega : \mathbb{X} \rightarrow \mathbb{Y}$ (e.g. a neural network). This means that instead of relying on a single set of weights, we average predictions

over all plausible parameter values according to their posterior probability, naturally capturing model uncertainty.

2.2.1 Uncertainty Decomposition

A key advantage of Bayesian modeling is that it enables a principled treatment of uncertainty in predictions. In Scientific Machine Learning, quantifying uncertainty is essential, as models are often used not only for prediction but also for decision-making, control, and discovery of physical laws. Importantly, not all sources of uncertainty have the same origin or interpretation. It is therefore useful to distinguish between *aleatoric* and *epistemic* uncertainty.

Given a Bayesian neural network with posterior $p(\omega \mid \mathcal{D})$, the predictive distribution for a new input $\mathbf{y}_\star \notin \mathcal{D}$ is:

$$p(\mathbf{y}_\star \mid \mathcal{D}) = \int p(\mathbf{y}_\star \mid \omega) p(\omega \mid \mathcal{D}) d\omega. \quad (14)$$

This distribution captures all uncertainty about the prediction $\mathbf{y}_\star$. Its total variance can be decomposed using the law of total variance as

$$\text{Var}(\mathbf{y}_\star \mid \mathcal{D}) = \mathbb{E}_{p(\omega \mid \mathcal{D})} [\text{Var}(\mathbf{y}_\star \mid \omega)] + \text{Var}_{p(\omega \mid \mathcal{D})} (\mathbb{E}[\mathbf{y}_\star \mid \omega]). \quad (15)$$

The first term,

$$\mathbb{E}_{p(\omega \mid \mathcal{D})} [\text{Var}(\mathbf{y}_\star \mid \omega)], \quad (16)$$

is known as *aleatoric uncertainty*. It arises from inherent randomness or noise in the data-generating process, such as measurement noise, stochastic physical effects, or unresolved microscopic degrees of freedom. Aleatoric uncertainty cannot be reduced by collecting more data, as it reflects irreducible variability in the system.

The second term,

$$\text{Var}_{p(\omega \mid \mathcal{D})} (\mathbb{E}[\mathbf{y}_\star \mid \omega]), \quad (17)$$

is referred to as *epistemic uncertainty*. It quantifies uncertainty in the model parameters due to limited data or model misspecification. Unlike aleatoric uncertainty, epistemic uncertainty can, in principle, be reduced by acquiring more informative data or by improving the model class.

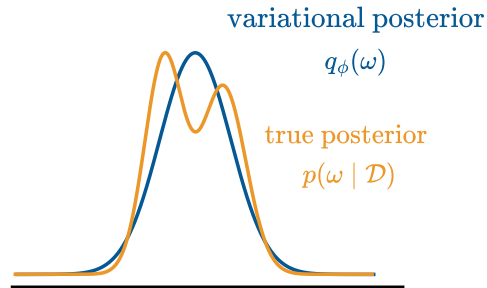
This distinction is particularly important in the context of SciML. In many scientific applications, such as climate modeling or molecular dynamics, one must differentiate between uncertainty stemming from intrinsic physical variability and uncertainty due to lack of knowledge about the governing processes. Epistemic uncertainty highlights regions of the input space where the model is poorly informed and can guide adaptive data acquisition or experimental design. Aleatoric uncertainty, on the other hand, reflects physical noise or unresolved scales and is critical for assessing the reliability of predictions in stochastic systems.

2.2.2 Prior Selection and Scalability

At first glance, Bayesian inference seems conceptually simple: one chooses a prior distribution, observes data, and updates beliefs using Bayes' theorem. In practice, however, applying this framework to modern NNs is extremely challenging. A key difficulty lies in specifying an appropriate prior (Wilson and Izmailov 2020; Vladimirova et al. 2019; Fortuin 2022). The prior strongly influences the posterior distribution and, consequently, the model's inductive behavior. Its choice is therefore far from arbitrary and raises important considerations around expressiveness, regularization, and robustness—issues emphasized in classical consistency results (Doob 1949). In this work, we will develop specific priors in the context of Scientific Machine Learning.

An even deeper challenge is computing the posterior itself. The Bayesian update requires evaluating a normalizing constant, which involves integrating the product $p(\mathcal{D} \mid \omega)p(\omega)$ over the entire parameter space. For modern NNs, this space is extremely high-dimensional and often exhibits a complex, multimodal structure, making the integral analytically intractable and computationally expensive. Classical approaches that provide asymptotically exact posterior inference, like Markov Chain Monte Carlo (MCMC) methods (Brooks et al. 2011), are theoretically appealing but often impractical for deep networks due to the high dimensionality of parameter spaces, slow mixing, and the need for careful tuning of step sizes. Gradient-based variants, such as Langevin dynamics or stochastic gradient Langevin dynamics (SGLD) (Welling and Teh 2011), or Stochastic Gradient Markov Chain Monte Carlo (SGMCMC) (Nemeth and Fearnhead 2021; T. Chen et al. 2014), improve scalability by using gradient

Figure 2: Variational inference approximation. Bayesian posterior estimation is recast as an optimisation problem by approximating the true posterior $p(\omega \mid \mathcal{D})$ with a parameterised distribution $q_\phi(\omega)$. The parameters ϕ are chosen to minimise their divergence from the true posterior.



information to guide exploration while injecting noise to preserve the correct stationary distribution. These methods can efficiently navigate complex, multi-modal landscapes and are compatible with mini-batch training, making them more suitable for Deep Learning. However, they still require careful hyperparameter tuning (e.g., step size and noise scale), and convergence guarantees are typically asymptotic, which may limit their practical fidelity to the true posterior in very high-dimensional or highly non-convex settings. As a result, exact Bayesian inference is generally infeasible in Deep Learning, necessitating the use of approximate methods.

2.2.3 Variational Inference

Exact Bayesian inference in neural networks is typically intractable because the posterior distribution over parameters, $p(\omega \mid \mathcal{D})$, involves a high-dimensional integral that cannot be computed analytically. *Variational inference* (Jordan et al. 1999; Wainwright, Jordan, et al. 2008; Graves 2011) offers a practical solution by turning inference into an optimization problem (Figure 2). The idea is to approximate the true posterior with a simpler, tractable distribution $q_\phi(\omega)$, parameterized by ϕ , such as a Gaussian or a more expressive distribution (Louizos and Welling 2016; Louizos and Welling 2017; Rudner et al. 2022).

The goal of variational inference is to find the parameters ϕ such that $q_\phi(\omega)$ is as close as possible to the true posterior. This is formalized by minimizing the KL divergence between the approximate and true posterior:

$$D_{\text{KL}}[q_\phi(\omega) \parallel p(\omega \mid \mathcal{D})] = \mathbb{E}_{q_\phi(\omega)} [\log q_\phi(\omega) - \log p(\omega \mid \mathcal{D})]. \quad (18)$$

Since the true posterior $p(\omega \mid \mathcal{D})$ is not accessible, this KL divergence cannot be computed directly. However, it can be shown that minimizing this KL divergence is equivalent to maximizing the Evidence Lower Bound (ELBO):

$$\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(\omega)} [\log p(\mathcal{D} \mid \omega)] - \text{KL}[q_\phi(\phi) \parallel p(\omega)]. \quad (19)$$

We can derive the above easily, starting from the definition of the KL divergence between the approximate posterior $q_\phi(\omega)$ and the true posterior $p(\omega \mid \mathcal{D})$:

$$\text{KL}[q_\phi(\omega) \parallel p(\omega \mid \mathcal{D})] = \mathbb{E}_{q_\phi(\omega)} [\log q_\phi(\omega) - \log p(\omega \mid \mathcal{D})]. \quad (20)$$

Using Bayes' theorem, we get:

$$\text{KL}[q_\phi(\omega) \parallel p(\omega \mid \mathcal{D})] = \mathbb{E}_{q_\phi(\omega)} \left[\log q_\phi(\omega) - \log \frac{p(\mathcal{D} \mid \omega)p(\omega)}{p(\mathcal{D})} \right]. \quad (21)$$

Rearranging terms gives

$$\log p(\mathcal{D}) \geq \log p(\mathcal{D}) - \text{KL}[q_\phi(\omega) \parallel p(\omega \mid \mathcal{D})] = \text{ELBO}(\phi). \quad (22)$$

Thus, minimizing the KL divergence is equivalent to maximizing the ELBO. This of two terms: the expected log-likelihood encourages the approximate posterior to explain the observed data well, while the KL term keeps it close to the prior, preventing overfitting.

Variational inference is computationally efficient and scales to large datasets, but its accuracy depends on the choice of the approximating family q_ϕ . Restrictive choices, such as simple unimodal Gaussians, may bias the approximation and fail to capture the full complexity of the true posterior (Blei et al. 2017). Recent advances, including stochastic variational inference (Hoffman et al. 2013) and amortized inference (Rezende and S. Mohamed 2015; Diederik P Kingma and Welling 2013), address these limitations by allowing the approximate posterior to be more flexible and by using shared neural network-based parameterizations to speed up and scale inference.

2.2.4 Example of Bayesian Neural Networks

In Bayesian Neural Networks, uncertainty is represented through a posterior distribution over network parameters $p(\omega \mid \mathcal{D})$. As discussed above, this posterior is generally intractable for modern deep architectures, motivating the development of approximate inference methods. These methods differ in how they approximate the posterior and in the computational compromises they introduce.

VARIATIONAL BAYESIAN NEURAL NETWORKS. Early practical Bayesian neural networks were developed using variational inference with factorized approximate posteriors and explicit sampling of network weights during training (Blundell et al. 2015). In this framework, the posterior is usually approximated by a factorized (over layers l) Gaussian:

$$p(\omega \mid \mathcal{D}) \approx q_\phi(\omega) = \prod_l \mathcal{N}(\omega_l \mid \mu_l, \text{diag}(\sigma_l^2)), \quad (23)$$

where $\phi = \{\mu, \sigma\}$. Predictions are obtained by sampling weights $\omega \sim q_\phi(\omega)$ and performing forward passes through the network. While this approach is conceptually faithful to the Bayesian formulation, sampling full weight matrices during training leads to high-variance gradient estimates, making training unstable, and substantial computational overhead.

MONTE CARLO DROPOUT. An alternative approximation is provided by Monte Carlo Dropout (Gal and Ghahramani 2016). This method interprets dropout regularization as approximate variational inference in a deep Gaussian process. Rather than treating each weight as a single point estimate, MC Dropout defines an approximate posterior over the weights of each layer l as

$$p(y_\star \mid \mathcal{D}) \approx q(\omega) = \prod_{l \geq 1} q(\omega_l), \quad \omega_l = \theta_l \odot \eta_l, \quad \eta_l \sim \text{Bernoulli}(p_l), \quad (24)$$

where θ_l are the (deterministic) variational parameters — i.e., the standard learned weight matrices — and η_l is a binary dropout mask applied element-wise. In other words, sampling from $q(\omega_l)$ is equivalent to drawing a dropout

mask and applying it to the weights, so the network’s stochasticity at training and test time is the variational distribution.

This is paired with a simple, factorized isotropic Gaussian prior over the weights,

$$p(\omega) = \prod_{l \geq 1} p(\omega_l) = \prod_{l \geq 1} \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}). \quad (25)$$

Variational inference seeks $q(\omega)$ minimizing the KL divergence to the true posterior. The key result of (Gal and Ghahramani 2016) is that, under the Bernoulli/Gaussian pair above, the KL term admits a closed-form Gaussian approximation whose gradient with respect to the variational parameters θ_l is

$$\frac{\partial}{\partial \theta_l} D_{\text{KL}}[q(\omega_l) \parallel p(\omega_l)] = \frac{(1 - p_l)}{2\sigma^2} \theta_l, \quad (26)$$

i.e., the KL term itself contributes $\frac{(1-p_l)}{2\sigma^2} \|\theta_l\|^2$ to the loss. This is precisely the form of standard *L2 weight decay*, with effective decay coefficient $\lambda_l = (1 - p_l)/(2\sigma^2)$ scaled by the retention probability of the layer. Consequently, a network trained with ordinary dropout and L2 regularization is *already* optimizing a variational lower bound.

At test time, dropout is kept active, and multiple stochastic forward passes are performed, yielding samples

$$\omega^{(t)} \sim q(\omega), \quad (27)$$

where $q(\omega)$ is the implicit distribution induced by the random dropout masks defined above. The predictive distribution is approximated by Monte Carlo integration over T such samples,

$$p(y_\star \mid \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^T p(y_\star \mid \omega^{(t)}). \quad (28)$$

Monte Carlo Dropout requires no modification of standard training pipelines and provides a lightweight approximation to Bayesian model averaging. However, the resulting posterior approximation is constrained by the dropout architecture — it is discrete (a mixture of Bernoulli-gated point masses rather than a continuous density) — and does not explicitly capture correlations

between parameters, since $q(\omega)$ factorizes independently across layers and units.

DEEP ENSEMBLES. Among the approximate Bayesian methods, Ensemble methods (Lakshminarayanan et al. 2017; Maddox et al. 2019; G. Huang et al. 2017) are the most famous ones due to simplicity and strong performance. Rather than constructing an explicit approximation to the posterior distribution, they approximate the Bayesian predictive distribution directly by training multiple deterministic NNs with different random initializations and, optionally, different data orderings or subsets. This procedure can be interpreted as approximating BMA (Wilson and Izmailov 2020; Izmailov et al. 2021) through sampling different modes of the loss landscape induced by stochastic optimization. While straightforward to implement and often highly effective in practice, ensembles come at the cost of substantial computational and memory overhead, especially when the number of models grows or when each model is itself large.

2.2.5 *Scaling Bayesian Neural Networks by Local Reparameterization*

A major bottleneck of naive stochastic variational inference for Bayesian neural networks is the way weight uncertainty is sampled. In the standard reparameterization trick, a single noise vector ϵ is drawn per weight matrix ω and shared across all examples in a minibatch. Because every datapoint in the batch is then multiplied by the *same* sampled weights, the resulting gradient estimates of the ELBO are highly correlated across the batch, making training slow and unstable, especially for large weight matrices.

To mitigate this limitation, the local reparameterization trick (Durk P Kingma et al. 2015) was introduced to sample the *pre-activations* of a Bayesian linear layer directly, rather than sampling its weights. The key observation is that a linear transformation of Gaussian-distributed weights is itself Gaussian: instead of sampling ω and then computing $z = h\omega$, one can compute the induced distribution over z analytically and sample from it directly. This is possible because, under a factorized (mean-field) Gaussian posterior over the weights, the pre-activations of different units in the batch become *conditionally*

independent given the input, converting global weight noise into local, per-example noise.

Concretely, consider a single linear layer with input $\mathbf{h} \in \mathbb{R}^M$ and weight matrix $\boldsymbol{\omega} \in \mathbb{R}^{M \times K}$, where each entry follows a factorized Gaussian posterior

$$\omega_{ij} \sim \mathcal{N}(\theta_{ij}, \alpha \theta_{ij}^2), \quad (29)$$

with θ_{ij} the variational mean and α a (shared) multiplicative variance parameter. Since the output $\mathbf{z} = \mathbf{h}\boldsymbol{\omega}$ is a linear combination of independent Gaussians, each output element z_j is itself Gaussian,

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\gamma}, \text{diag}(\boldsymbol{\delta})), \quad (30)$$

with mean and variance obtained analytically by propagating the moments of $\boldsymbol{\omega}$ through the linear map:

$$\gamma_j = \sum_i h_i \theta_{ij}, \quad \delta_j = \alpha \sum_i h_i^2 \theta_{ij}^2. \quad (31)$$

Rather than sampling the weights and propagating them forward, one can therefore sample the pre-activations directly using the standard reparameterization trick applied at the level of $\mathbf{z}$:

$$z_j = \gamma_j + \sqrt{\delta_j} \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, 1). \quad (32)$$

Two benefits follow directly from this reformulation. First, **statistical efficiency**: because a fresh noise variable ϵ_j is drawn independently for each unit and each example in the minibatch, the resulting stochastic gradients are decorrelated across the batch, and their variance now decreases proportionally to the minibatch size B rather than to the number of weight samples T . This yields substantially lower-variance gradient estimates for the same computational budget, translating into faster and more stable convergence of the ELBO. Second, **computational efficiency**: the trick avoids ever materializing a sampled $M \times K$ weight matrix per example. Instead, only the $O(B \times K)$ pre-activation statistics γ_j, δ_j need to be computed and sampled, which can be implemented with two ordinary matrix multiplications ($\mathbf{h}\boldsymbol{\theta}$ and $\mathbf{h}^2\boldsymbol{\theta}^2$) that are trivially parallelizable on GPU hardware, in contrast to naively sampling and applying a different weight matrix per datapoint in the batch.

Together, these properties make the local reparameterization trick a practical mechanism for scaling variational Bayesian neural networks to the sizes typically used in deep learning, without sacrificing the calibrated uncertainty estimates that motivate the Bayesian treatment in the first place.

BAYESIAN AUTOREGRESSIVE AND RECURRENT NEURAL NETWORKS

Autoregressive and recurrent networks have achieved remarkable progress across various fields, from weather forecasting to molecular generation and Large Language Models. Despite their strong predictive capabilities, these models lack a rigorous framework for addressing uncertainty, which is key in scientific applications such as PDE solving, molecular generation, and Machine Learning Force Fields. We propose BARNN, a variational Bayesian Autoregressive and Recurrent Neural Network that provides a principled way to turn any autoregressive or recurrent model into its Bayesian counterpart. BARNN is based on the variational dropout method, making it applicable to large recurrent architectures, and we further introduce a temporal version of the Variational Mixtures of Posteriors prior (tVAMP-prior) to make Bayesian inference efficient and well-calibrated. Extensive experiments on PDE modelling and molecular generation demonstrate that BARNN not only achieves comparable or superior predictive accuracy but also excels in UQ and modelling long-range dependencies.

3.1 INTRODUCTION

Autoregressive and recurrent models have demonstrated impressive advancements across a wide range of disciplines, from weather prediction (Bodnar et al. 2025) to molecule generation (Shi et al. 2020; Simm et al. 2020) and Large Language Models (LLMs) (Bengio, Ducharme, and Vincent 2000; Vaswani et al. 2017; Radford et al. 2018). Despite their strong predictive capabilities, these models are prone to overfitting to the specific tasks they are trained on,

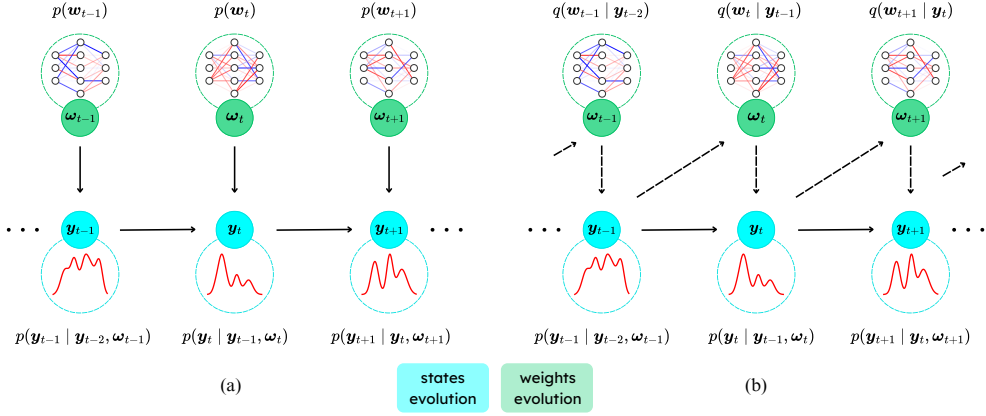


Figure 3: BARNN generative and inference process. Solid lines show the generative process (a), while dotted lines show the variational inference process (b). Network weights w_t evolve jointly with the states y_t , forming a coupled probabilistic model.

limiting their applicability outside the training domain (Papamarkou et al. 2024). Overcoming this behaviour is critical, not only in scientific applications where phenomena can exhibit complex distributional shifts away from the training data, but also in deep over-parametrized models. In weather forecasting, for instance, rapidly changing climate patterns can cause significant deviations from the training distribution and lead to unpredictable results, while over-parametrized LLMs often produce incorrect answers with high confidence, highlighting fundamental issues with model calibration (Z. Jiang et al. 2021; Xiao et al. 2022; A. X. Yang et al. 2022). This problem is prevalent across many other domains, and Bayesian approaches represent a promising direction for addressing it (Papamarkou et al. 2024).

This chapter aims to bridge the gap between autoregressive and recurrent models and Bayesian methods, presenting a fully Bayesian, scalable, calibrated, and accurate framework named Bayesian Autoregressive and Recurrent Neural Network (BARNN). BARNN provides a principled way to convert any autoregressive or recurrent model into its Bayesian counterpart. In BARNN, the network weights evolve jointly with the observable states (e.g. language tokens, PDE states), forming a joint probabilistic model (see Figure 3). A new state is generated by sampling from the distribution conditioned on previous

states and the current network weights, while the latter are drawn from a variational posterior distribution conditioned on the history of observed states.

We derive a variational lower bound for efficiently training BARNN, related to the VAE (Diederik P Kingma and Welling 2013) objective (ELBO). To make Bayesian inference over a large number of network weights computationally tractable, we propose an extension of Variational Dropout (Durk P Kingma et al. 2015; Gal and Ghahramani 2016), which also opens the door to network quantization and efficiency (Louizos and Welling 2017; Van Baalen et al. 2020) for over-parametrized autoregressive models. Furthermore, we introduce a temporal version of the Variational Mixtures of Posteriors prior (Tomczak and Welling 2018) (tVAMP-prior) to make Bayesian inference both efficient and well-calibrated. Building on the recent successful application of autoregressive and recurrent models to SciML tasks (Pfaff et al. 2021; Lippe et al. 2024; Segler et al. 2018), we apply our framework to UQ for neural PDE solvers and to *molecule generation* with language models for drug discovery. Experiments carried out in this chapter demonstrate that BARNN not only achieves greater predictive accuracy, but also provides calibrated and sharp uncertainty estimates, and excels at modelling long-range molecular dependencies compared to related approaches. To the best of our knowledge, BARNN is the first framework that converts any autoregressive or recurrent model into its Bayesian version with minimal architectural modifications, yielding improved accuracy over its non-Bayesian counterpart and offering a structured approach to UQ.

3.2 BACKGROUND AND RELATED WORK

3.2.1 *Autoregressive and Recurrent Networks*

Autoregressive and recurrent models represent a joint probability distribution as a product of conditional distributions, leveraging the probability chain rule (Bengio, Ducharme, and Vincent 2000; Uria et al. 2016). Over recent years, these models have been successfully applied to a broad range of tasks, including text generation (Bengio, Ducharme, and Vincent 2000; Vaswani et al. 2017), graph modelling (Yujia Li et al. 2018; Q. Liu et al. 2018), audio synthesis (Dieleman et al. 2016), and, more recently, PDE modelling (Brandstetter, Welling, et al. 2022; Brandstetter, Worrall, et al. 2022) and molecule

generation (Segler et al. 2018; Özçelik et al. 2024; Schmidinger et al. 2024). Despite their wide applicability, these models are known to overfit, producing unreliable predictions particularly in over-parametrized regimes – such as modern LLMs – or when the test distribution shifts significantly from the training distribution (Papamarkou et al. 2024). Yet quantifying the uncertainty in the model parameters, namely *epistemic uncertainty*, remains far from standard practice, and principled methods to do so are still largely missing from the literature (Aichberger et al. 2024).

3.2.2 Neural PDE Solvers

Surrogate modelling is a rapidly growing area of research in which deep learning models, referred to as *Neural PDE Solvers*, are trained to approximate solutions to complex physical phenomena (Z. Li, Kovachki, et al. 2020; Bhattacharya et al. 2021; Rozza et al. 2022; Pichi et al. 2024; Coscia, Demo, et al. 2024; Z. Li, H. Zheng, et al. 2024). Among these, autoregressive Neural PDE Solvers (Brandstetter, Worrall, et al. 2022; Sanchez-Gonzalez et al. 2020) have attracted considerable attention, as they can infer solutions to temporal PDEs orders of magnitude faster than classical solvers and produce longer stable rollouts than their non-autoregressive counterparts. A known limitation, however, is that errors accumulate at each autoregressive step (Brandstetter, Worrall, et al. 2022), gradually shifting the effective data distribution and degrading solution quality over long horizons. Several approaches have been proposed to quantify this growing uncertainty: PDE Refiner (Lippe et al. 2024) obtains PDE solutions by iteratively refining a latent variable initialised from Gaussian noise, in the spirit of denoising diffusion models (Ho et al. 2020); while GraphCast (Lam et al. 2022) introduced Input Perturbation, which injects small Gaussian noise into the initial condition and aggregates predictions across multiple rollouts to derive uncertainty estimates.

3.2.3 Recurrent Neural Networks for Molecule Generation

Drug design centres on the discovery of novel molecules capable of binding effectively to specific biomolecular targets. Generative deep learning has

emerged as a powerful tool for this task (Shi et al. 2020; Eijkelboom et al. 2024), with Chemical Language Models (CLMs) – in which molecules are represented as molecular strings – offering a particularly promising direction. Recurrent neural networks have shown significant potential in this setting (Segler et al. 2018; Özçelik et al. 2024; Schmidinger et al. 2024). In the experimental section of this chapter, we demonstrate how equipping classical RNNs with the BARNN framework enhances performance, improving the model’s ability to capture long-range dependencies, generating statistically more robust molecules, and achieving more faithful learning of molecular properties.

3.3 METHODS

The Bayesian Autoregressive and Recurrent Neural Network (BARNN) provides a principled and accessible way to convert any autoregressive or recurrent model into its Bayesian counterpart. The framework constructs a joint distribution over observable states (e.g. language tokens, PDE states) and model weights, with both evolving alternately in time (Section 3.3.1). We show how to train the model by deriving a novel variational lower bound that connects naturally to the VAE framework (Diederik P Kingma and Welling 2013) (Section 3.3.2). Finally, we present a scalable variational posterior and prior for efficient weight-parameter sampling (Section 3.3.3).

3.3.1 The State-Weight Model

Autoregressive models represent a joint probability distribution as a product of factorised conditional distributions over *states*:

$$p(\mathbf{y}_0, \dots, \mathbf{y}_T) = \prod_{t=1}^T p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0), \quad (33)$$

where a new state $\mathbf{y}_t$ is sampled from a distribution conditioned on all previous states $\mathbf{y}_{t-1}, \dots, \mathbf{y}_0$, and T denotes the trajectory length. Deep autoregressive models (Schmidhuber, Hochreiter, et al. 1997) approximate the factorised distribution $p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0)$ with a neural network parameterised by optimisable deterministic weights θ .

We construct a principled Bayesian extension of this framework by jointly modelling states $\mathbf{y}_t$ and stochastic weights ω_t through a *joint* distribution $p(\mathbf{y}_{0:T}, \omega_{1:T})$, accounting for the temporal variability of both¹. The full joint distribution takes the form:

$$p(\mathbf{y}_{0:T}, \omega_{1:T}) = \prod_{t=1}^T p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0, \omega_t) p(\omega_t). \quad (34)$$

A new state $\mathbf{y}_t$ is thus obtained by sampling from the distribution conditioned on all previous states and the current weights ω_t , while the latter are drawn from a prior distribution $p(\omega_t)$, as illustrated in Figure 3.

3.3.2 The Temporal Variational Lower Bound

Training the model in Equation (34) requires maximising the log-likelihood $\log p(\mathbf{y}_{\geq 0})$. Directly optimising $\log p(\mathbf{y}_{0:T})$ by marginalising Equation (34) over the weights, or via expectation maximisation, is intractable; doing so requires evaluating the integral

$$p(\mathbf{y}_{0:T}) = \int p(\mathbf{y}_{0:T} \mid \omega_t) p(\omega_t) d\omega_t, \quad (35)$$

which has no closed-form solution because the likelihood $p(\mathbf{y}_{0:T} \mid \omega_t)$ is a highly non-linear, non-conjugate function of the weights distribution, and the integral is high-dimensional to be approximated numerically. Consequently, the true posterior over the weights,

$$p(\omega_t \mid \mathbf{y}_{0:T}) = \frac{p(\mathbf{y}_{0:T} \mid \omega_t) p(\omega_t)}{p(\mathbf{y}_{0:T})}, \quad (36)$$

is equally intractable, as its normalising constant is precisely the marginal likelihood above. We therefore optimise $\log p(\mathbf{y}_{0:T})$ through variational inference (Diederik P Kingma and Welling 2013). The variational posterior over network weights conditioned on the states, $q_\phi(\omega_{1:T} \mid \mathbf{y}_{0:T})$, is parameterised by time-independent weights ϕ and factorises as:

$$q_\phi(\omega_{1:T} \mid \mathbf{y}_{0:T}) = \prod_{t=1}^T q_\phi(\omega_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0). \quad (37)$$

¹ We denote a sequence $(a_k, a_{k+1}, \dots, a_{k+l})$ as $a_{k:k+l} \forall l > k \in \mathbb{N}$.

Given this variational posterior, in Appendix 3.6.1 we derive the following variational lower bound to be maximised over the variational parameters:

$$\mathcal{L}(\phi) = \mathbb{E}_{t \sim \mathcal{U}[1, T]} \left[\mathbb{E}_{\omega_t \sim q_\phi} [\log p(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t)] - D_{\text{KL}} [q_\phi(\omega_t \mid \mathbf{y}_{0:t-1}) \parallel p(\omega_t)] \right]. \quad (38)$$

This objective has two notable properties. First, it mirrors the VAE objective (Diederik P Kingma and Welling 2013), establishing a connection between Bayesian networks and latent variable models. Second, BARNN incorporates explicit timestep dependency into the ELBO, allowing the weights to adapt over time – a property we show leads to sharper, better-calibrated uncertainty estimates and improved modelling of long-range dependencies. In the limiting case of a time-independent peaked distribution with prior and posterior perfectly aligned, the BARNN objective reduces to the standard log-likelihood used in conventional autoregressive and recurrent models (see Appendix 3.6.5). This suggests interpreting Equation (38) as a form of Bayesian weight regularisation during autoregressive training. Once the model is trained, the predictive distribution and uncertainty estimates are readily obtained via Monte Carlo sampling (see Appendix 3.6.4).

3.3.3 Variational Dropout Approximation

A central practical challenge in both the generative model of Equation (34) and the lower bound of Equation (38) is how to sample network weights efficiently. Direct sampling is computationally prohibitive for large networks, making alternative strategies necessary. We adopt Variational Dropout (VD) (Durk P Kingma et al. 2015; Molchanov et al. 2017) and the *local reparameterisation trick* (see Section 2.2.4) to sample activations rather than weights directly, yielding significant computational savings. We extend VD to handle dynamic weights ω_t during both training and inference, and we call it time Variational Adaptive Dropout (tVAD). Specifically, we seek a reparameterisation $\omega_t = f(\theta, \alpha_t, \epsilon)$, where θ are static deterministic network weights, α_t are deterministic time-dependent variational coefficients, $\epsilon \sim p(\epsilon)$ is random noise, and f is a differentiable function. This parameterisation offers three key advantages: (i) it admits the global reparameterisation trick (Diederik P Kingma and Welling 2013); (ii) it relies on fixed weights θ , enabling the approach to scale to large

networks; and (iii) it introduces temporal and state dependency by perturbing θ with time-dependent coefficients α_t and stochasticity via ϵ , requiring minimal modifications to the underlying autoregressive or recurrent architecture. We describe one concrete instantiation of this reparameterisation below, leaving alternative formulations for future work.

VARIATIONAL POSTERIOR. We assume that the weights ω_t factorise across layers $l \in \{1, \dots, L\}$ under the posterior q_ϕ :

$$q_\phi(\omega_t \mid y_{0:t-1}) = \prod_{l=1}^L q_\phi(\omega_t^l \mid y_{0:t-1}), \quad (39)$$

where ω_t^l denotes the weights at time t for layer l . We further assume that each $q_\phi(\omega_t^l \mid y_{0:t-1})$ follows a Gaussian distribution:

$$\begin{aligned} q_\phi(\omega_t^l \mid y_{0:t-1}) &= \mathcal{N}(\alpha_t^l \theta^l, (\alpha_t^l \theta^l)^2) \\ &\stackrel{\Rightarrow}{=} \\ \omega_t^l &= \alpha_t^l \theta^l (1 + \epsilon), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}), \end{aligned} \quad (40)$$

where the square on θ^l is applied component-wise, implying that weights within each layer are uncorrelated, and α_t^l are scalar positive dropout coefficients that depend on $y_{0:t-1}$ (and possibly on t depending on the application). In practice, an inference network E_ψ outputs the full vector of dropout coefficients across all layers, $\alpha_t = [\alpha_t^1, \dots, \alpha_t^L] = E_\psi(y_{0:t-1})$, and the local reparameterisation trick (Durk P Kingma et al. 2015) is applied component-wise for linear layers. The variational parameters to optimise are therefore $\phi = (\theta, \psi)$, with $\theta = \{\theta^1, \dots, \theta^L\}$ the static weights and ψ the inference network weights.

AGGREGATED VARIATIONAL POSTERIOR IN TIME PRIOR. To identify a suitable prior, we observe that if the KL divergence in Equation (38) is independent of θ , then maximising $\mathcal{L}$ with respect to θ for fixed α_t is equivalent to maximising the expected log-likelihood $\log p(y_t \mid \theta, \psi, y_{0:t-1})^2$. We therefore seek a prior under which the KL term is independent of θ .

² This is equivalent to optimising the weights θ in a standard deterministic network using maximum likelihood estimation.

Following the approach of (Tomczak and Welling 2018), we find that the optimal prior for the lower bound in Equation (38) is given by (proof in Appendix 3.6.2):

$$p(\omega_t) = \int p(\mathbf{y}_{0:t-1}) q(\omega_t | \mathbf{y}_{0:t-1}) d\mathbf{y}_{0:t-1}. \quad (41)$$

We refer to this prior as the temporal Variational Mixture of Posteriors (tVAMP). It is related to the VAMP prior of (Tomczak and Welling 2018), but differs in two important respects: (i) VAMP is a prior over latent variables, whereas tVAMP is a prior over Bayesian network weights; and (ii) tVAMP is time-dependent, with the optimal prior at each time t given by the aggregated posterior, a temporal structure absent from (Tomczak and Welling 2018).

Assuming the prior factorises analogously to the posterior, the computations (see Appendix 3.6.2) yield the following closed-form prior, where k indexes the batch:

$$\begin{aligned} p(\omega_t^l) &= \mathcal{N}(\beta_t^l \theta^l, (\gamma_t^l \theta^l)^2), \\ \beta_t^l &= \frac{1}{N} \sum_{k=1}^N \alpha_t^l(\mathbf{y}_{0:t-1}^k), \\ \gamma_t^l &= \sqrt{\frac{1}{N} \sum_{k=1}^N (\alpha_t^l(\mathbf{y}_{0:t-1}^k))^2} \quad \forall l, t. \end{aligned} \quad (42)$$

For the prior and posterior defined above, the KL divergence is indeed independent of θ (proof in Appendix 3.6.3) and takes the form:

$$D_{\text{KL}} [q_\phi(\omega_t | \mathbf{y}_{0:t-1}) \| p(\omega_t)] = \sum_{l=1}^L \frac{|\theta^l|}{2} \left[\left(\frac{\alpha_t^l - \beta_t^l}{\gamma_t^l} \right)^2 + \left(\frac{\alpha_t^l}{\gamma_t^l} \right)^2 - 1 - 2 \ln \frac{\alpha_t^l}{\gamma_t^l} \right], \quad (43)$$

where $|\theta^l|$ denotes the number of parameters in layer l .

3.3.4 Bayesian Neural PDE Solvers

Autoregressive Neural PDE Solvers map PDE states $\mathbf{y}_t$ to future states $\mathbf{y}_{t+1}$ given an initial condition $\mathbf{y}_0$ (Brandstetter, Worrall, et al. 2022). These solvers

are deterministic and produce no estimates of uncertainty. BARNN converts any such solver into its Bayesian counterpart through a small number of targeted modifications. We model the joint weight-state distribution as:

$$p(\mathbf{y}_{0:T}, \boldsymbol{\omega}_{1:T}) = \prod_{t=1}^T p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \boldsymbol{\omega}_t) p(\boldsymbol{\omega}_t), \quad (44)$$

which is a special case of Equation (34) under a first-order Markov assumption on the states. We adopt a Gaussian state distribution $p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \boldsymbol{\omega}_t) = \mathcal{N}(\mathbf{N0}(\mathbf{y}_{t-1}; \boldsymbol{\omega}_t), \text{diag}(\sigma_t^2))$, where the mean at time t is produced by the Neural PDE Solver applied to $\mathbf{y}_{t-1}$, and the standard deviation is fixed rather than learned. Here $\mathbf{N0}$ denotes *any* Neural PDE Solver architecture with probabilistic weights $\boldsymbol{\omega}_t$, confirming that the framework is architecture-agnostic. The probabilistic weights are obtained via the layerwise transformation $f(\boldsymbol{\theta}, \boldsymbol{\alpha}_t, \epsilon)$ described in Equation (40), with the inference network E_ψ receiving a single state as input under the Markov assumption. Training proceeds by maximising Equation (38), which reduces to minimising the standard one-step MSE loss (Brandstetter, Welling, et al. 2022) augmented by a Bayesian weight regularisation term given by the negative KL divergence in Equation (43). The full training procedure is detailed in Appendix 3.9.

3.3.5 Bayesian Recurrent Neural Networks

Given a sequence $\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_T$, a standard Recurrent Neural Network (RNN) (Bengio, Ducharme, and Vincent 2000; Schmidhuber, Hochreiter, et al. 1997) evaluates Equation (33) by maintaining hidden variables $\{\mathbf{h}_t\}_{t=0}^T$ that accumulate³ input information up to time t . The conditional distribution is modelled as:

$$\begin{aligned} p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0) &= \sigma(\mathbf{h}_{t-1}), \\ \mathbf{h}_t &= \text{NN}(\mathbf{y}_t, \mathbf{h}_{t-1}; \boldsymbol{\theta}), \end{aligned} \quad (45)$$

where σ is the softmax function, NN is a neural network with deterministic weights $\boldsymbol{\theta}$, and the specific gating mechanism depends on the RNN architecture. The hidden variable $\mathbf{h}_t$ summarises the history $\mathbf{y}_{t-1}, \dots, \mathbf{y}_0$ up to time t .

³ With the convention $\mathbf{h}_{-1} = \mathbf{0}$.

Extending RNNs to their Bayesian counterpart within BARNN is straightforward. The state distribution becomes:

$$\begin{aligned} p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0, \boldsymbol{\omega}_t) &= \sigma(\mathbf{h}_{t-1}), \\ \mathbf{h}_t &= \text{NN}(\mathbf{y}_t, \mathbf{h}_{t-1}; \boldsymbol{\omega}_t), \end{aligned} \quad (46)$$

while the variational posterior over the weights is conditioned on the hidden state:

$$q_\psi(\boldsymbol{\omega}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0) = q_\psi(\boldsymbol{\omega}_t \mid \mathbf{y}_{t-1}, \mathbf{h}_{t-2}). \quad (47)$$

The probabilistic weights $\boldsymbol{\omega}_t$ are obtained via the layerwise transformation $f(\boldsymbol{\theta}, \boldsymbol{\alpha}_t, \boldsymbol{\epsilon})$ from Equation (40), with the inference network E_ψ taking $\mathbf{y}_{t-1}$ and $\mathbf{h}_{t-2}$ as input. Training proceeds by maximising Equation (38), which reduces to minimising the cross-entropy loss (Bengio, Ducharme, and Vincent 2000) standard in causal language modelling, augmented by the negative KL divergence in Equation (43). The training procedure is reported in Appendix 3.9.

3.4 EXPERIMENTS

We demonstrate the effectiveness of BARNN across a range of tasks. We begin by validating the tVAMP prior on a synthetic time series dataset, showing its advantage over the widely used log-uniform prior. We then apply BARNN to Neural PDE Solvers (Bar-Sinai et al. 2019; Brandstetter, Worrall, et al. 2022; Z. Li, Kovachki, et al. 2020) to assess both solution accuracy and UQ. Finally, we apply BARNN to RNNs for unconditional molecule generation (Segler et al. 2018; Özçelik et al. 2024) using the SMILES syntax, and show improved generation quality compared to existing methods. Code is publicly available at <https://github.com/dario-coscia/barnn>. Please refer to Appendix 3.7 for datasets, metric, model and training specifics.

Table 1: Prior ablation on sinusoidal time-series forecasting. RMSE, NLL, and ECE for different BARNN priors, Monte Carlo Dropout, and deterministic baselines. Static denotes a baseline in which the initial condition is used as the prediction at every time step.

Model	Prior	MSE ($\downarrow$)	NLL ($\downarrow$)	ECE ($\downarrow$)
Static	-	$0.490^{\pm 0.000}$	-	-
MLP	-	$0.081^{\pm 0.011}$	-	-
Dropout (p=0.5)	-	$0.072^{\pm 0.004}$	$0.593^{\pm 0.461}$	$0.084^{\pm 0.010}$
Dropout (p=0.2)	-	$0.048^{\pm 0.004}$	$-0.075^{\pm 0.004}$	$0.068^{\pm 0.009}$
BARNN	log-uniform	$0.045^{\pm 0.003}$	$-0.092^{\pm 0.064}$	$0.050^{\pm 0.016}$
BARNN	tVAMP	$0.043^{\pm 0.001}$	$-0.166^{\pm 0.019}$	$0.049^{\pm 0.008}$

3.4.1 Time Series

We assess the tVAMP prior on a synthetic time series forecasting task. The dataset consists of sinusoidal signals with varying frequencies and phases:

$$\begin{cases} x_t = x_{t-1} + \frac{3\pi}{100}, & x_0 = 0, \\ y_t = \frac{1}{5} \sum_{j=1}^5 \sin(\alpha_j x_t + \beta_j), \end{cases} \quad (48)$$

with $\alpha_i \sim U[0.5, 1.5]$, $\beta_i \sim U[0, 3\pi]$, and $t \in \{1, \dots, 100\}$. This controlled setting isolates the effect of prior choice on forecasting accuracy and uncertainty calibration. We generate 1024 trajectories for training and 100 for testing. All models share the same architecture – a 2-layer neural network with 64 units and ReLU activations – and are trained for 1500 epochs using the Adam optimiser (Diederik P Kingma 2014) with learning rate 10^{-4} and weight decay 10^{-8} .

TVAMP OUTPERFORMS THE LOG-UNIFORM PRIOR Table 1 compares different prior choices for BARNN against Monte Carlo Dropout (Gal and Ghahramani 2016) and a deterministic MLP baseline. BARNN with the tVAMP prior achieves the best performance across all metrics. Compared to the log-uniform prior, tVAMP yields a lower NLL, reflecting better-calibrated predictive uncertainty, as well as the lowest RMSE and ECE. These results confirm that the

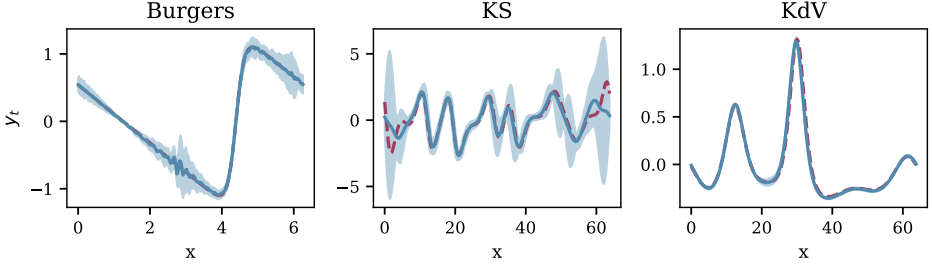


Figure 4: Prediction and uncertainty intervals at the final time step. CFD solution (red dotted line), BARNN mean prediction (solid blue line), and ± 3 standard-deviation confidence region (shaded blue) for each PDE.

tVAMP prior improves both forecasting accuracy and uncertainty estimation in sequential models.

3.4.2 PDE Modelling

PDE modelling consists of learning solution operators that generalise across instances within a given equation family. We evaluate BARNN on three benchmark PDEs – *Burgers*, *Kuramoto-Sivashinsky* (KS), and *Korteweg-de Vries* (KdV) (L. C. Evans 2022; Brandstetter, Welling, et al. 2022) – and compare against Monte Carlo Dropout (Gal and Ghahramani 2016), Variational Dropout with Empirical Bayes (ARD) (Kharitonov et al. 2018), Input Perturbation (Lam et al. 2022) (Perturb), and PDE Refiner (Lippe et al. 2024) (Refiner). Full details on dataset generation, architectures, baselines, and hyperparameters are provided in Appendix 3.7, alongside additional results.

BARNN SOLVES PDES WITH CALIBRATED UNCERTAINTIES Trained Neural Solvers are unrolled for 320 steps and evaluated via negative log-likelihood (NLL) and expected calibration error (ECE). NLL jointly rewards prediction accuracy and sharpness of the predictive distribution, while ECE measures alignment between estimated and empirical probabilities. Table 2 shows that BARNN achieves the lowest NLL and ECE across all three PDEs. Notably, despite models exhibiting broadly comparable RMSE values, their NLL diverges

Table 2: Uncertainty quantification for Neural PDE Solvers. NLL and ECE for Burgers, KS, and KdV over 320 autoregressive steps. Mean and std over four random seeds; *break* denotes divergence.

Model	NLL ($\downarrow$)			ECE ($\downarrow$)		
	Burgers	KS	KdV	Burgers	KS	KdV
BARNN	-2.51 ± 0.29	-0.87 ± 0.52	-0.94 ± 1.23	0.05 ± 0.03	0.04 ± 0.03	0.10 ± 0.01
ARD	5.00 ± 7.11	10.7 ± 11.6	break	0.26 ± 0.10	0.17 ± 0.01	0.19 ± 0.05
Dropout	0.06 ± 4.59	-0.48 ± 0.65	0.40 ± 4.57	0.17 ± 0.15	0.10 ± 0.01	0.14 ± 0.07
Perturb	-2.01 ± 0.08	-0.65 ± 1.07	-0.37 ± 1.32	0.16 ± 0.01	0.05 ± 0.02	0.11 ± 0.06
Refiner	break	break	break	0.30 ± 0.08	0.23 ± 0.02	0.34 ± 0.09

substantially (Figure 5), a consequence of underestimated ensemble variance in competing methods. This finding underscores that pointwise RMSE alone is insufficient for evaluating Neural PDE Solvers in uncertainty-aware settings, as low prediction error can coexist with poorly calibrated uncertainty estimates.

ADAPTIVE CONFIDENCE INTERVALS Figure 4 illustrates BARNN predictions alongside 99.7% confidence intervals across different spatial regions. BARNN dynamically adjusts interval width in response to local uncertainty: intervals widen appropriately in high-uncertainty regions, such as those governed by the KS equation, and narrow where the model closely tracks the true solution, as in the KdV case. This adaptive behaviour demonstrates that BARNN produces uncertainty estimates that are both meaningful and geometrically consistent with the underlying PDE dynamics.

CONVERGENCE WITH ENSEMBLE SIZE A desirable property of ensemble-based methods is the convergence of statistical moments as the number of ensemble members grows. As shown in Figure 6, BARNN achieves convergent RMSE with approximately 30 ensemble members, confirming that reliable moment estimates require only a modest number of additional stochastic forward passes. When uncertainty estimates are not needed, using the maximum a posteriori (MAP) weight estimate reduces inference to a single forward pass, matching the computational cost of standard deterministic methods.

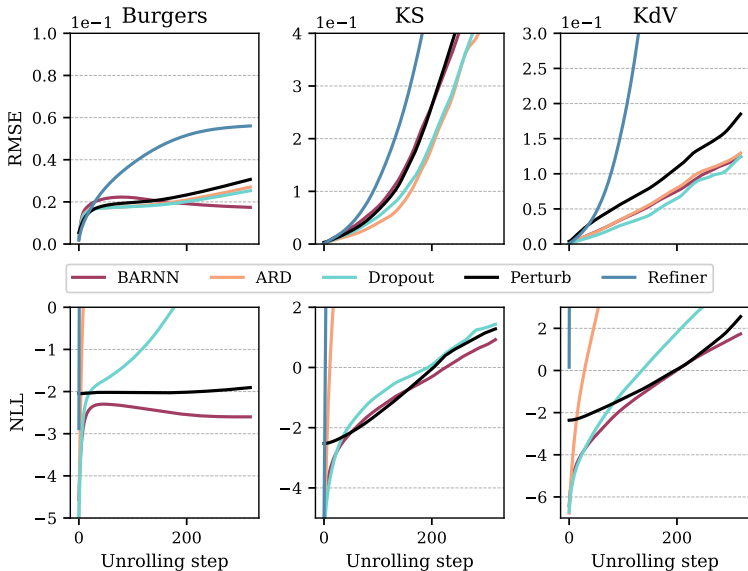


Figure 5: RMSE and NLL across Neural PDE Solvers. Despite comparable RMSE, NLL varies substantially across methods due to underestimated ensemble variance.

3.4.3 Molecule Generation

Molecular generation requires navigating a vast chemical space while respecting long-range structural dependencies within molecular graphs. We evaluate BARNN on unconditional molecule generation by representing molecules in SMILES notation and pre-training an LSTM (Schmidhuber, Hochreiter, et al. 1997) on the ChEMBL dataset from (Özçelik et al. 2024), comprising approximately 1.9M SMILES strings. We compare against a standard LSTM (Schmidhuber, Hochreiter, et al. 1997) and the SMILES LSTM of (Segler et al. 2018), which augments the standard LSTM with dropout applied to both input and hidden variables. Further details on the dataset, metrics, hyperparameters, and baselines are provided in Appendix 3.8.

BARNN CORRECTLY LEARNS THE SMILES SYNTAX BARNN achieves the highest validity, diversity, novelty, and uniqueness across all baselines (Table 3), demonstrating that the Bayesian framework does not compromise generative

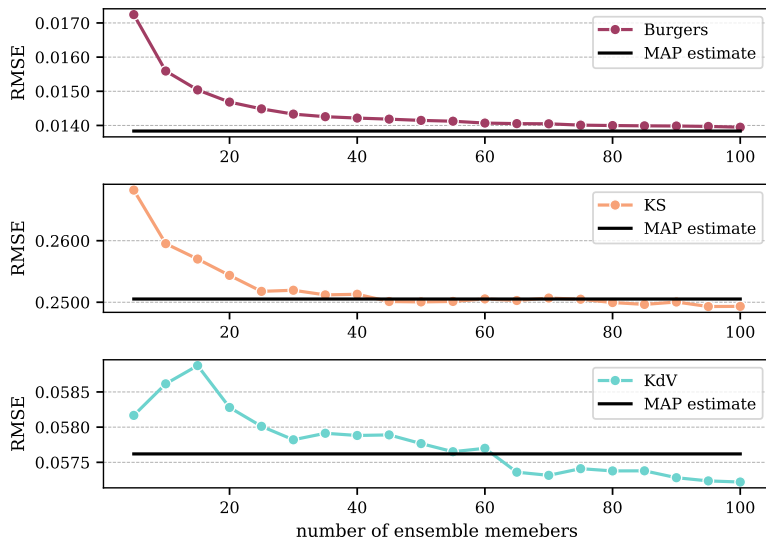


Figure 6: RMSE convergence with ensemble size in BARNN. Accurate moment estimates are obtained with a small number of stochastic forward passes; a single MAP forward pass suffices when uncertainty is not required.

quality. Gains are consistent across all metrics, with diversity showing only marginal improvement, as expected given the diversity of the underlying training corpus.

LONG-RANGE DEPENDENCIES IN SMILES We sampled 50K SMILES strings from each model and analysed the sources of error in invalid outputs (Figure 7a). The dominant failure mode across all models is incorrect ring assignment – specifically, opening a ring without closing it. BARNN substantially reduces this type of error compared to both baselines. In SMILES notation, ring closures are encoded by matching numerical labels on bonded atoms, requiring the model to remember an open ring label across potentially many intervening tokens – a prototypical long-range dependency. As shown in Figure 7b, validity degrades for all models as ring count increases, but BARNN maintains higher validity throughout, confirming its superior ability to track long-range structural constraints.

Table 3: De novo drug-like molecule generation. Validity, diversity, novelty, and uniqueness for BARNN and baselines. Mean and std over four random seeds.

Model	Validity ($\uparrow$)	Diversity ($\uparrow$)	Novelty ($\uparrow$)	Uniqueness ($\uparrow$)
BARNN	95.09 ± 0.34	88.63 ± 0.01	95.41 ± 0.24	95.06 ± 0.34
SMILES LSTM	94.60 ± 0.27	88.57 ± 0.09	94.25 ± 0.15	94.58 ± 0.28
LSTM	93.02 ± 0.33	88.57 ± 0.02	92.81 ± 0.34	92.14 ± 0.15

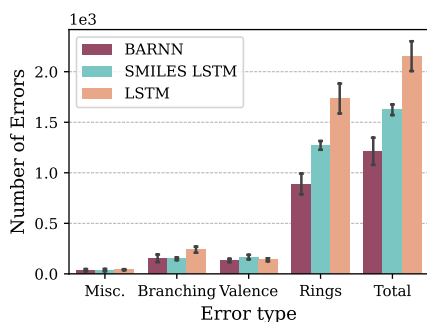
Table 4: Molecular property distribution matching. Wasserstein distance W_2 between generated and training molecular property distributions. Mean and std over four random seeds.

Model	W_2 ($\downarrow$)					
	MW	HBDs	HBAs	Rotable Bonds	LogP	TPSA
BARNN	2.53 ± 1.03	0.02 ± 0.01	0.05 ± 0.03	0.13 ± 0.03	0.06 ± 0.01	0.90 ± 0.38
SMILES LSTM	10.4 ± 3.71	0.03 ± 0.02	0.15 ± 0.06	0.14 ± 0.05	0.11 ± 0.05	1.81 ± 0.97
LSTM	12.3 ± 2.91	0.03 ± 0.01	0.20 ± 0.07	0.13 ± 0.07	0.12 ± 0.03	1.84 ± 1.04

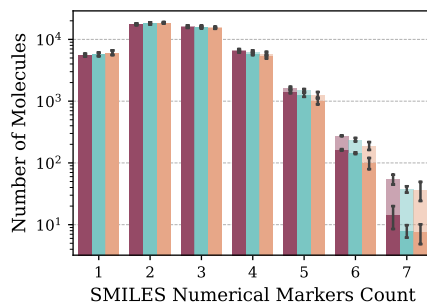
BARNN ACCURATELY REPRODUCES MOLECULAR PROPERTIES We measure the Wasserstein distance W_2 between the property distributions of generated and training molecules across six descriptors: molecular weight (MW), H-bond donors (HBDs), H-bond acceptors (HBAs), rotatable bonds, LogP, and topological polar surface area (TPSA). BARNN achieves the lowest W_2 across all properties (Table 4), with particularly pronounced improvements on molecular weight – the hardest property to match – where competing methods show substantially larger distributional discrepancies. Taken together, the results in Tables 3 and 4 establish that BARNN not only generates more statistically valid molecules but also more faithfully reproduces the distributional characteristics of the training data.

3.5 FINAL DISCUSSION

We have introduced BARNN, a scalable, calibrated, and accurate framework for converting any autoregressive or recurrent model into its Bayesian counter-



(a) Error sources for invalid SMILES. Ring assignment errors dominate across all models.



(b) Validity as a function of ring count. Shaded bars include all generated SMILES; solid bars show only valid ones.

Figure 7: SMILES error analysis. (a) Sources of invalid outputs and (b) validity as a function of ring count. Mean and standard deviation over four random-seed initialisations.

part, bridging the gap between sequential modelling and principled Bayesian inference. BARNN constructs a joint probabilistic model by evolving network weights alongside observable states, supported by a novel variational lower bound for efficient training, an extension of Variational Dropout for dynamic weight sampling, and a temporal VAMP-prior to enhance calibration.

We validated BARNN on autoregressive Neural PDE Solvers and RNN-based molecule generation. In PDE modelling, BARNN surpasses existing methods in both accuracy and uncertainty calibration, requiring only a small ensemble for reliable moment estimates and reducing to a single forward pass when uncertainty is not needed. In molecule generation, BARNN produces outputs with higher validity, diversity, novelty, and uniqueness than its non-Bayesian counterparts, while better capturing long-range dependencies and molecular property distributions.

As a general and flexible framework, BARNN can be readily extended to other sequential domains such as text, audio, and time series. Promising directions for future work include the design of more informative priors that incorporate domain knowledge, the investigation of dynamic pruning at each

autoregressive step for large over-parametrized models, and a deeper analysis of the interplay between epistemic and aleatoric uncertainty over time.

CHAPTER APPENDIX

3.6 PROOFS AND DERIVATIONS

This appendix collects the formal proofs supporting the results presented in the main text. Section 3.6.1 establishes the variational lower bound, Section 3.6.2 derives the tVAMP prior, and Section 3.6.3 contains the KL divergence computation between the prior and posterior introduced in Section 3.3.3.

3.6.1 Variational Lower Bound

We aim to optimise the log-likelihood for the model:

$$p(\mathbf{y}_{0:T}, \boldsymbol{\theta}_{1:T}) = \prod_{t=1}^T p(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \boldsymbol{\omega}_t) p(\boldsymbol{\omega}_t). \quad (49)$$

Introducing the variational posterior:

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{1:T} \mid \mathbf{y}_{0:T}) = \prod_{t=1}^T q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t \mid \mathbf{y}_{0:t-1}), \quad (50)$$

we derive the following evidence lower bound:

$$\begin{aligned}
\log p(\mathbf{y}_{0:T}) &= \log \int p(\mathbf{y}_{0:T}, \boldsymbol{\theta}_{1:T}) d\boldsymbol{\theta}_{1:T} \\
&= \log \int p(\mathbf{y}_{0:T}, \boldsymbol{\theta}_{1:T}) \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{1:T} | \mathbf{y}_{0:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{1:T} | \mathbf{y}_{0:T})} d\boldsymbol{\theta}_{1:T} \\
&\geq \mathbb{E}_{\boldsymbol{\theta}_{1:T} \sim q_{\boldsymbol{\phi}}} \left[\log \frac{p(\mathbf{y}_{0:T}, \boldsymbol{\theta}_{1:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{1:T} | \mathbf{y}_{0:T})} \right] \\
&= \mathbb{E}_{\boldsymbol{\theta}_{1:T}} \left[\log \prod_{t=1}^T \frac{p(\mathbf{y}_t | \mathbf{y}_{0:t-1}, \boldsymbol{\omega}_t) p(\boldsymbol{\theta}_t)}{q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t | \mathbf{y}_{0:t-1})} \right] \\
&= \sum_{t=1}^T \mathbb{E}_{\boldsymbol{\omega}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t | \mathbf{y}_{0:t-1})} \left[\log \frac{p(\mathbf{y}_t | \mathbf{y}_{0:t-1}, \boldsymbol{\omega}_t) p(\boldsymbol{\theta}_t)}{q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t | \mathbf{y}_{0:t-1})} \right] \\
&= \sum_{t=1}^T \left\{ \mathbb{E}_{\boldsymbol{\omega}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t | \mathbf{y}_{0:t-1})} [\log p(\mathbf{y}_t | \mathbf{y}_{0:t-1}, \boldsymbol{\omega}_t)] - D_{\text{KL}} [q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t | \mathbf{y}_{0:t-1}) \| p(\boldsymbol{\theta}_t)] \right\} \\
&= \sum_{t=1}^T \mathcal{L}_{\text{cumulative}}(\boldsymbol{\phi}, t).
\end{aligned} \tag{51}$$

The Bayesian treatment naturally yields a sum of per-timestep evidence lower bounds, meaning that maximising the model likelihood requires maximising the *cumulative* ELBO. For scalable training, we approximate this quantity following the convention in causal language modelling:

$$\mathcal{L}(\boldsymbol{\phi}) \approx \mathbb{E}_{t \sim \mathcal{U}[1, T]} [\mathcal{L}_{\text{cumulative}}(\boldsymbol{\phi}, t)]. \tag{52}$$

3.6.2 Temporal Variational Mixture of Posteriors

We seek the prior that maximises the evidence lower bound:

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{\omega}_{1:T} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_{1:T} | \mathbf{y}_{0:T})} \left[\log \frac{p(\mathbf{y}_{0:T}, \boldsymbol{\omega}_{1:T})}{q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_{1:T} | \mathbf{y}_{0:T})} \right].$$

Optimising over the prior yields:

$$\begin{aligned}
p^*(\omega_{1:T}) &= \arg \max_{p(\omega_{1:T})} \int \mathcal{L}(\phi) p(\mathbf{y}_{0:T}) d\mathbf{y}_{0:T} \\
&= \arg \max_{p(\omega_{1:T})} \int p(\mathbf{y}_{0:T}) q(\omega_{1:T} | \mathbf{y}_{0:T}) \log p(\omega_{1:T}) d\mathbf{y}_{0:T} d\omega_{1:T} \\
&= \arg \max_{p(\omega_{1:T})} \int \left[\int p(\mathbf{y}_{0:T}) q(\omega_{1:T} | \mathbf{y}_{0:T}) d\mathbf{y}_{0:T} \right] \log p(\omega_{1:T}) d\omega_{1:T} \\
&= \arg \max_{p(\omega_{1:T})} -H \left[\int p(\mathbf{y}_{0:T}) q(\omega_{1:T} | \mathbf{y}_{0:T}) d\mathbf{y}_{0:T} | p(\omega_{1:T}) \right] \\
&= \int p(\mathbf{y}_{0:T}) q(\omega_{1:T} | \mathbf{y}_{0:T}) d\mathbf{y}_{0:T},
\end{aligned} \tag{53}$$

where H denotes cross entropy, minimised when the two distributions coincide. Marginalising to each time t :

$$\begin{aligned}
p^*(\omega_t) &= \int p^*(\omega_{1:T}) d\omega_{\neq t} \\
&= \int p(\mathbf{y}_{0:T}) q(\omega_{1:T} | \mathbf{y}_{0:T}) d\mathbf{y}_{0:T} d\omega_{\neq t} \\
&= \int p(\mathbf{y}_{0:t-1}) q(\omega_t | \mathbf{y}_{0:t-1}) d\mathbf{y}_{0:t-1}.
\end{aligned} \tag{54}$$

This result extends (Tomczak and Welling 2018) in two respects: (i) the prior here is over Bayesian network weights rather than latent variables; and (ii) it is time-dependent, with the optimal prior at each t given by the aggregated posterior, a structure absent from (Tomczak and Welling 2018).

Assuming the prior factorises analogously to the variational posterior:

$$\begin{aligned}
p^*(\omega_t) &\approx \prod_{l=1}^L \prod_{i,j_l} \int p(\mathbf{y}_{0:t-1}) \mathcal{N}(\alpha_t^l \theta_{i,j_l}^l, (\alpha_t^l \theta_{i,j_l}^l)^2) d\mathbf{y}_{0:t-1} \\
&= \prod_{l=1}^L \prod_{i,j_l} \mathcal{N}(\beta_t^l \theta_{i,j_l}^l, (\gamma_t^l \theta_{i,j_l}^l)^2),
\end{aligned} \tag{55}$$

where β_t^l and γ_t^l are obtained as:

$$\begin{aligned}\beta_t^l \theta_{ijl}^l &= \mathbb{E}[\theta_{ijl}^l] \\ &= \theta_{ijl}^l \int \alpha_t^l(\mathbf{y}_{0:t-1}) p(\mathbf{y}_{0:t-1}) d\mathbf{y}_{0:t-1} \\ &\approx \theta_{ijl}^l \frac{1}{N} \sum_{k=1}^N \alpha_t^l(\mathbf{y}_{0:t-1}^k).\end{aligned}\tag{56}$$

$$\begin{aligned}(\gamma_t^l \theta_{ijl}^l)^2 &= \mathbb{V}\text{ar}[\theta_{ijl}^l] \\ &= (\theta_{ijl}^l)^2 \int (\alpha_t^l(\mathbf{y}_{0:t-1}))^2 p(\mathbf{y}_{0:t-1}) d\mathbf{y}_{0:t-1} \\ &\approx (\theta_{ijl}^l)^2 \frac{1}{N} \sum_{k=1}^N (\alpha_t^l(\mathbf{y}_{0:t-1}^k))^2.\end{aligned}\tag{57}$$

In summary:

$$\beta_t^l = \frac{1}{N} \sum_{k=1}^N \alpha_t^l(\mathbf{y}_{t-1}^k, t), \quad \gamma_t^l = \sqrt{\frac{1}{N} \sum_{k=1}^N (\alpha_t^l(\mathbf{y}_{t-1}^k, t))^2} \quad \forall l, t.\tag{58}$$

3.6.3 KL Divergence

We compute $D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t \mid \mathbf{y}_{0:t-1}) \parallel p(\boldsymbol{\omega}_t)]$. By factorisation across layers:

$$D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t \mid \mathbf{y}_{0:t-1}) \parallel p(\boldsymbol{\omega}_t)] = \sum_{l=1}^L D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t^l \mid \mathbf{y}_{0:t-1}) \parallel p(\boldsymbol{\omega}_t^l)].\tag{59}$$

Letting $\theta_{ij;t}^l$ denote the matrix entries of $\boldsymbol{\omega}_t^l$, and substituting the variational posterior $q_{\boldsymbol{\phi}}(\theta_{ij;t}^l \mid \mathbf{y}_{0:t-1}) = \mathcal{N}(\alpha_t^l \theta_{ij}^l, (\alpha_t^l \theta_{ij}^l)^2)$ and $p(\theta_{ij;t}^l) = \mathcal{N}(\beta_t^l \theta_{ij}^l, (\gamma_t^l \theta_{ij}^l)^2)$:

$$D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_t^l \mid \mathbf{y}_{0:t-1}) \parallel p(\boldsymbol{\omega}_t^l)] = \sum_{i,j,l} \frac{1}{2} \left[\left(\frac{\alpha_t^l - \beta_t^l}{\gamma_t^l} \right)^2 + \left(\frac{\alpha_t^l}{\gamma_t^l} \right)^2 - 1 - 2 \ln \left(\frac{\alpha_t^l}{\gamma_t^l} \right) \right],\tag{60}$$

where positivity of the dropout rates was used in the final simplification. Combining Equations (60) and (59) concludes the proof. Notably, this KL divergence is independent of θ_{ij}^l for all i, j, l .

3.6.4 Predictive Distribution and Uncertainty Estimates

This section describes how BARNN is used at inference time for both prediction and UQ, decomposing uncertainty into its epistemic and aleatoric components.

PREDICTIVE DISTRIBUTION. The model predictive distribution $p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0)$ is obtained by marginalising over weights sampled from the variational posterior:

$$p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0) = \mathbb{E}_{\omega_t \sim q_\phi} [p(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t) q_\phi(\omega_t \mid \mathbf{y}_{0:t-1})]. \quad (61)$$

In practice, this is approximated by averaging over multiple stochastic forward passes (Gal and Ghahramani 2016; Louizos and Welling 2017).

UNCERTAINTY DECOMPOSITION. The predictive variance admits a clean decomposition via the law of total variance (Depeweg et al. 2018):

$$\sigma_t^2 = \underbrace{\sigma_{\omega_t}^2(\mathbb{E}[\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t])}_{\text{epistemic uncertainty}} + \underbrace{\mathbb{E}_{\omega_t}[\sigma^2(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t)]}_{\text{aleatoric uncertainty}}, \quad (62)$$

with $\omega_t \sim q_\phi(\omega_t \mid \mathbf{y}_{0:t-1})$, and $\mathbb{E}[\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t]$, $\sigma^2(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t)$ the mean and variance of $p(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \dots, \mathbf{y}_0, \omega_t)$ respectively.

MONTE CARLO ESTIMATES. Both the predictive distribution in Equation (61) and the uncertainty decomposition in Equation (62) are approximated via Monte Carlo. For a fixed initial state $\mathbf{y}_0$, we unroll the model D times, each time drawing independent weights ω_t . Letting $\mu_{t,i}$ and $\sigma_{t,i}^2$ denote the mean and variance of $p(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \omega_t)$ for the i -th sample, the unbiased estimates are:

$$\mu_t = \frac{1}{D} \sum_{i=1}^D \mu_{t,i}, \quad \sigma_t^2 = \underbrace{\frac{1}{D} \sum_{i=1}^D \mu_{t,i}^2 - \mu_t^2}_{\text{epistemic}} + \underbrace{\frac{1}{D} \sum_{i=1}^D \sigma_{t,i}^2}_{\text{aleatoric}}. \quad (63)$$

3.6.5 Connection to Deterministic Autoregressive Models

We show that the BARNN objective in Equation (38) reduces to the standard log-likelihood under two simplifying assumptions:

$$(a.1) \quad \omega_t = \omega \quad \forall t, \quad (\text{static weights})$$

$$(a.2) \quad q(\omega \mid \mathbf{y}_{0:t-1}) = p(\omega) = \delta(\omega - \theta) \quad \forall t. \quad (\text{deterministic weights})$$

Under (a.2), the KL divergence vanishes; under (a.1), the expectation collapses to a single value. The bound then reduces to:

$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}[1, T]} [\log p(\mathbf{y}_t \mid \mathbf{y}_{0:t-1}, \theta)], \quad (64)$$

which is precisely the log-likelihood optimised in standard autoregressive training.

3.7 BARNN FOR PDE MODELLING ADDITIONAL RESULTS

This section details the experimental setup for the PDE modelling experiments and provides additional results.

3.7.1 Data Generation and Metrics

DATA GENERATION. We consider three benchmark PDEs commonly used to evaluate Neural PDE Solvers (Brandstetter, Welling, et al. 2022; Brandstetter, Worrall, et al. 2022; Bar-Sinai et al. 2019):

$$\begin{aligned} \text{Burgers} \quad & \partial_t u + u \partial_x u - \nu \partial_{xx} u = 0, \\ \text{Kuramoto-Sivashinsky} \quad & \partial_t u + u \partial_x u + \partial_{xx} u + \partial_{xxxx} u = 0, \\ \text{Korteweg-de Vries} \quad & \partial_t u + u \partial_x u + \partial_{xxx} u = 0, \end{aligned} \quad (65)$$

where $\nu > 0$ is the viscosity coefficient. We follow the data generation protocol of (Brandstetter, Welling, et al. 2022), imposing periodic boundary conditions and sampling initial conditions from a truncated Fourier series $u^0(x) = \sum_{k=1}^{10} A_k \sin(2\pi l_k \frac{x}{L} + \eta_k)$, with random coefficients $\{A_k, \eta_k, l_k\}_k$ as in (Brandstetter, Welling, et al. 2022). Space-time discretisation parameters are reported in Table 5.

METRICS. We evaluate probabilistic solvers using three complementary metrics:

Table 5: PDE discretisation parameters. n_t and n_x : time and space step sizes; $t_{\max}$: simulation duration; L : domain length.

	PDE	n_t	n_x	$t_{\max}$	L
Train	<i>Burgers</i>	0.1	0.25	14	2π
	<i>KS</i>	0.1	0.25	14	64
	<i>KdV</i>	0.1	0.25	14	64
Test	<i>Burgers</i>	0.1	0.25	32	2π
	<i>KS</i>	0.1	0.25	32	64
	<i>KdV</i>	0.1	0.25	32	64

1. **Root Mean Square Error (RMSE)** (Brandstetter, Worrall, et al. 2022): measures agreement between ensemble mean predictions and ground truth.
2. **Negative Log-Likelihood (NLL)**: balances prediction accuracy against the sharpness of the predictive distribution.
3. **Expected Calibration Error (ECE)**: measures alignment between estimated and empirical probabilities.

Let μ_t, σ_t be the ensemble mean and standard deviation, and $\mathbb{Q}_p$ the Gaussian quantile function at level p . Then:

$$\begin{aligned}
 \text{RMSE} &= \sqrt{\frac{1}{n_x n_t} \sum_{x,t} [y_t(x) - \mu_t(x)]^2}, \\
 \text{NLL} &= \frac{1}{2n_x n_t} \sum_{x,t} \left[\log(2\pi\sigma_t^2(x)) + \frac{[y_t(x) - \mu_t(x)]^2}{\sigma_t^2(x)} \right], \\
 \text{ECE} &= \mathbb{E}_{p \sim \mathcal{U}(0,1)} [|p - p_{\text{obs}}|], \quad p_{\text{obs}} = \frac{1}{n_x n_t} \sum_{x,t} \mathbb{I} \{y_t(x) \leq \mathbb{Q}_p[\mu_t(x), \sigma_t^2(x)]\}.
 \end{aligned} \tag{66}$$

3.7.2 Architectures, Hyperparameters, and Computational Cost

All models were trained for 7000 epochs using the Adam optimiser with learning rate $5 \cdot 10^{-4}$ and weight decay 10^{-8} . Experiments were run on a single Quadro RTX 4000 GPU with 8 GB of memory, requiring approximately one day to train all models. Unless stated otherwise, uncertainty estimates are computed using 100 ensemble members and models are unrolled for 320 steps at inference.

The backbone architecture for all experiments is a Fourier Neural Operator (Z. Li, Kovachki, et al. 2020) with 8 layers of width 64, 32 Fourier modes, and swish activation (Z. Li, Kovachki, et al. 2020; Brandstetter, Welling, et al. 2022), totalling approximately 1 million trainable parameters. Batch normalisation was found to degrade uncertainty estimates and was excluded. A linear lifting layer and a linear projection layer are used at the input and output respectively. For dropout-based models (Dropout, ARD, and BARNN), the stochastic mask is applied only to the linear layers within the Fourier blocks and the final projection layer, as this configuration yielded the best-calibrated uncertainties.

DROPOUT. A dropout rate of 0.2 was applied to the Fourier linear layers. Alternative rates were explored but worsened calibration. Table 6 reports the ablation over dropout rates (single seed, due to computational cost).

INPUT PERTURBATION. The standard FNO was trained with Gaussian input noise (Lam et al. 2023): $\mathbf{u}_t^{\text{perturb}} = \mathbf{u}_t + \delta \max |\mathbf{u}_t| \epsilon$, with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ and $\delta = 0.01$. Table 7 reports an ablation over perturbation scales (single seed).

PDE REFINER. We implemented PDE Refiner following the pseudocode in (Lippe et al. 2024), and found the model sensitive to the number of refinement steps R and minimum variance $\sigma_{\min}$, consistent with prior observations (Kohl et al. 2024). We used $R = 2$ and $\sigma_{\min} = 2 \cdot 10^{-6}$. Table 8 reports the hyperparameter ablation (single seed).

Table 6: Dropout rate ablation. RMSE, NLL, and ECE on the validation set for 140-step unrolling across PDEs.

PDE	Dropout Rate	RMSE ($\downarrow$)	NLL ($\downarrow$)	ECE ($\downarrow$)
<i>Burgers</i>	0.2	0.014	-3.070	0.022
	0.5	0.030	-1.779	0.077
	0.8	0.055	-1.562	0.030
<i>KS</i>	0.2	0.044	-0.416	0.120
	0.5	0.066	-1.247	0.065
	0.8	0.183	0.911	0.127
<i>KdV</i>	0.2	0.017	-3.350	0.135
	0.5	0.040	-2.280	0.078
	0.8	0.080	-1.310	0.097

Table 7: Input perturbation scale ablation. RMSE, NLL, and ECE on the validation set for 140-step unrolling across PDEs.

PDE	Perturbation δ	RMSE ($\downarrow$)	NLL ($\downarrow$)	ECE ($\downarrow$)
<i>Burgers</i>	0.001	0.017	1.583	0.366
	0.01	0.014	-2.085	0.181
	0.1	0.093	0.218	0.193
<i>KS</i>	0.001	0.015	-1.048	0.188
	0.01	0.041	-1.942	0.110
	0.1	0.404	0.413	0.078
<i>KdV</i>	0.001	0.014	-0.641	0.231
	0.01	0.027	-2.032	0.111
	0.1	0.155	0.115	0.189

ARD DROPOUT. The model has no tunable hyperparameters; we follow (Kharitonov et al. 2018) for the implementation. A global dropout rate was used for a fair comparison with standard Dropout, applied only to the Fourier layers.

Table 8: PDE Refiner hyperparameter ablation. RMSE, NLL, and ECE on the validation set for 140-step unrolling across PDEs.

PDE	$\sigma_{\min}$	R	RMSE ($\downarrow$)	NLL ($\downarrow$)	ECE ($\downarrow$)
<i>Burgers</i>	$2 \cdot 10^{-6}$	2	0.019	44.18	0.216
	$2 \cdot 10^{-6}$	3	0.106	184.40	0.547
	$2 \cdot 10^{-6}$	4	0.021	16.64	0.196
	$2 \cdot 10^{-7}$	2	0.014	86.24	0.248
	$2 \cdot 10^{-7}$	3	0.026	240.40	0.239
	$2 \cdot 10^{-7}$	4	0.023	100.30	0.367
<i>KS</i>	$2 \cdot 10^{-6}$	2	0.027	6.089	0.167
	$2 \cdot 10^{-6}$	3	0.081	76.760	0.232
	$2 \cdot 10^{-6}$	4	0.041	8.456	0.179
	$2 \cdot 10^{-7}$	2	0.024	9.330	0.189
	$2 \cdot 10^{-7}$	3	0.025	6.977	0.183
	$2 \cdot 10^{-7}$	4	0.037	6.250	0.175
<i>KdV</i>	$2 \cdot 10^{-6}$	2	0.011	10.170	0.402
	$2 \cdot 10^{-6}$	3	0.031	72.100	0.343
	$2 \cdot 10^{-6}$	4	0.022	39.900	0.285
	$2 \cdot 10^{-7}$	2	0.043	86.740	0.356
	$2 \cdot 10^{-7}$	3	0.042	192.500	0.456
	$2 \cdot 10^{-7}$	4	0.042	92.330	0.321

BARNN. The dropout mask is applied exclusively to the fully connected layers within the Fourier blocks, consistent with the other dropout-based baselines. The dropout coefficients α_t are produced by an inference network E_ψ that takes as input the current state y_t and the time variable t . The state channels are projected to a fixed dimension of 16 via a linear layer, and the time variable is encoded using a sinusoidal positional encoding with period $T = 1000$ to modulate the state embedding. Four additional linear layers of width 16 with swish activation, followed by a sigmoid output layer, map the embedding to the dropout probabilities $p_{t'}$, from which dropout rates are

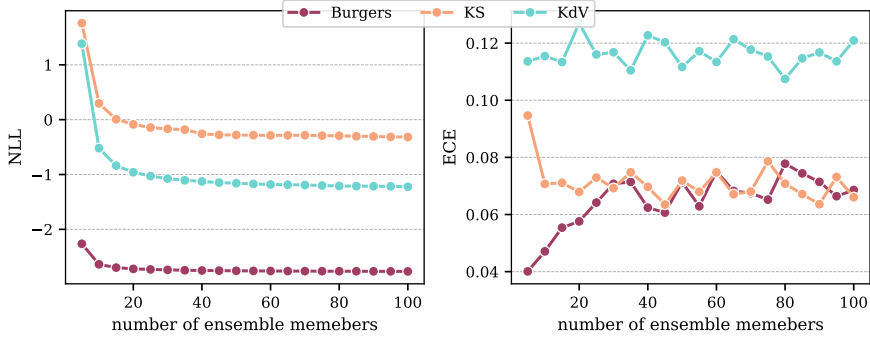


Figure 8: NLL and ECE convergence with ensemble size. Both metrics stabilise with a small number of ensemble members, consistent with the RMSE trend in Figure 6.

recovered as $p_t/(1 - p_t)$. The inference network contains fewer than 50K parameters, adding negligible overhead to training.

3.7.3 Additional Results

POINTWISE ACCURACY. Table 9 reports RMSE across all models and PDEs. Dropout-based methods (BARNN, ARD, and Dropout) consistently achieve the lowest errors, suggesting that Bayesian regularisation is beneficial not only for uncertainty calibration but also for predictive accuracy. Differences among the top-performing models are modest, and as discussed in Section 3.4.2, RMSE alone is insufficient for evaluating probabilistic solvers, since low prediction error can coexist with poorly calibrated uncertainties (see Figure 5).

ENSEMBLE SIZE CONVERGENCE. Figure 8 extends the analysis of Section 3.4.2 by reporting NLL and ECE as a function of ensemble size. Both metrics stabilise after approximately 30 members, mirroring the RMSE trend in Figure 6. Beyond this point, adding further members yields negligible improvement, confirming that BARNN produces well-calibrated and sharp uncertainty estimates with minimal computational overhead.

Table 9: Pointwise accuracy across Neural PDE Solvers. RMSE for Burgers, KS, and KdV over 320 autoregressive steps. Mean and std over four random seeds.

Model	RMSE ($\downarrow$)		
	<i>Burgers</i>	<i>KS</i>	<i>KdV</i>
BARNN	0.019 ± 0.004	0.217 ± 0.047	0.061 ± 0.026
ARD	0.020 ± 0.009	0.161 ± 0.052	0.063 ± 0.030
Dropout	0.019 ± 0.008	0.172 ± 0.026	0.053 ± 0.022
Perturb	0.022 ± 0.005	0.222 ± 0.067	0.090 ± 0.036
Refiner	0.042 ± 0.024	0.407 ± 0.402	0.775 ± 1.581

SOLUTION ROLLOUTS. Figures 9, 10, and 11 show representative one-dimensional rollouts over 320 steps using 100 ensemble members. As expected from the autoregressive nature of the solver, errors accumulate over time; however, the predictive variance grows correspondingly, reflecting the model’s ability to identify regions of increasing uncertainty. Despite the growing discrepancy, the predictive mean remains closely aligned with the ground truth, and the variance provides a reliable indicator of where the solution becomes less certain.

3.8 BARNN FOR MOLECULE GENERATION ADDITIONAL RESULTS

This section details the experimental setup for the molecule generation experiments and provides additional results.

3.8.1 Data Generation and Metrics

DATA GENERATION. Experiments focus on unconditional molecule generation using the SMILES notation. We use the dataset from (Özçelik et al. 2024), comprising approximately 1.9M SMILES strings extracted from the ChEMBL database (Zdrazil et al. 2024). SMILES strings were tokenised using a regular expression covering all elements and special SMILES characters (numbers, brackets, and so on), with each atom or character mapped to a single token.

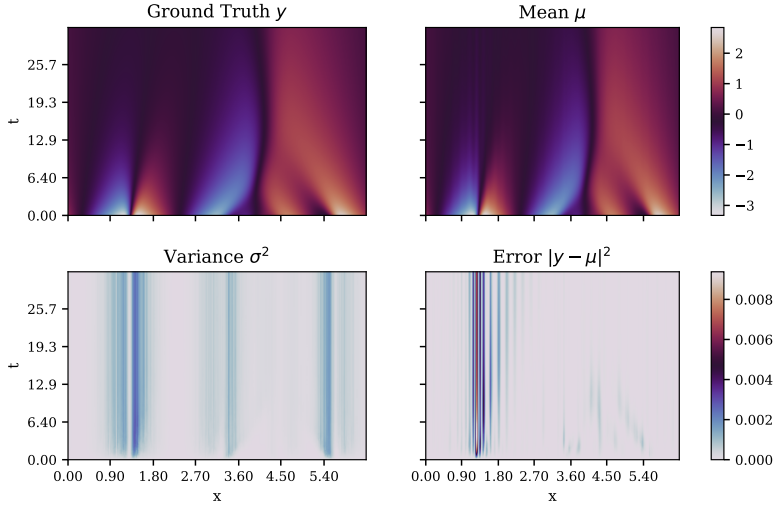


Figure 9: Burgers rollout. Representative one-dimensional solution over 320 autoregressive steps.

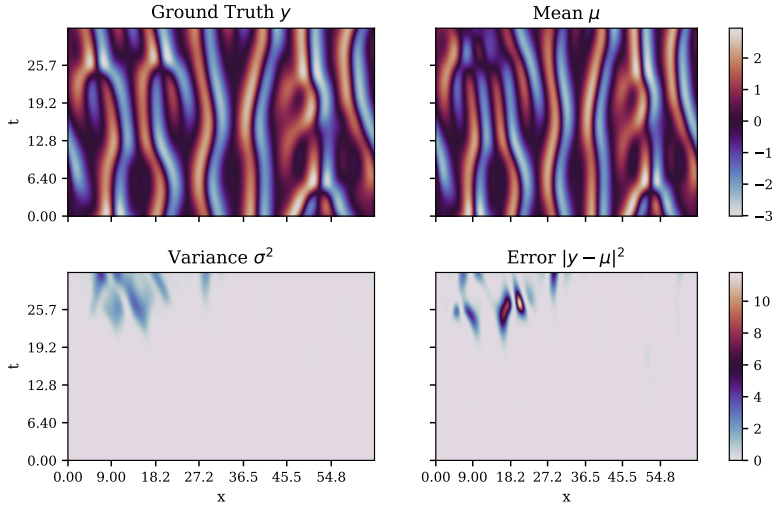


Figure 10: Kuramoto-Sivashinsky rollout. Representative one-dimensional solution over 320 autoregressive steps.

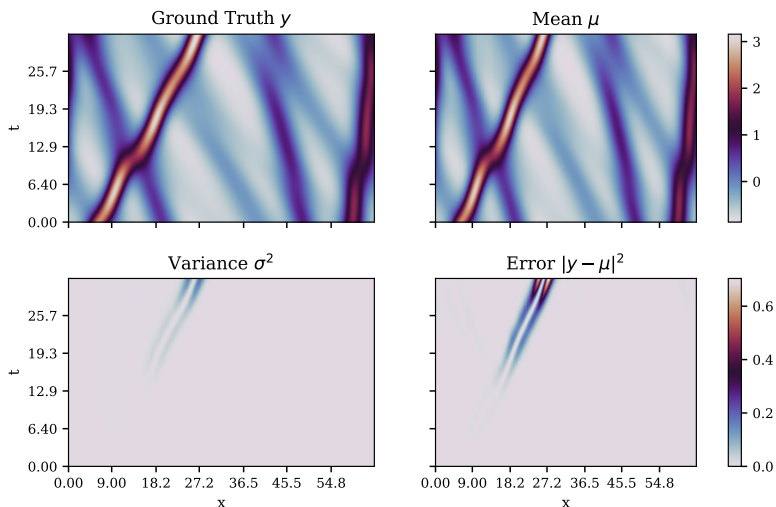


Figure 11: Korteweg-de Vries rollout. Representative one-dimensional solution over 320 autoregressive steps.

METRICS. Model performance is assessed using four generation quality metrics:

1. **Validity:** fraction of generated SMILES corresponding to chemically valid molecules.
2. **Diversity:** fraction of unique Murcko scaffolds among generated molecules.
3. **Novelty:** fraction of generated molecules absent from the training dataset.
4. **Uniqueness:** fraction of structurally unique molecules among all generated designs.

We additionally compute the Wasserstein distance between the molecular property distributions of generated molecules and the ChEMBL dataset, following (Segler et al. 2018), evaluating molecular weight, H-bond donors and acceptors, rotatable bonds, LogP, and TPSA.

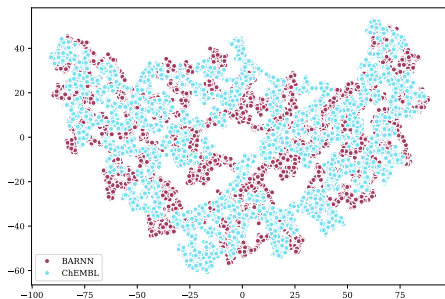


Figure 12: t-SNE of molecular descriptors. BARNN and ChEMBL distributions overlap substantially in the latent space.

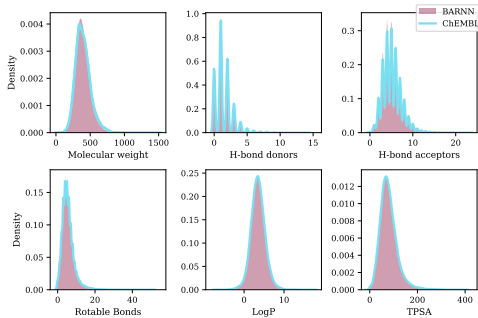


Figure 13: Per-descriptor property distributions. KDE plots for BARNN and ChEMBL molecules show near-perfect overlap across all descriptors.

3.8.2 Architectures, Hyperparameters, and Computational Cost

All models were trained for 12 epochs using the Adam optimiser with learning rate $2 \cdot 10^{-4}$ and weight decay 10^{-8} , distributed across four Tesla P100 GPUs with 16 GB memory each, using a batch size of 256. Training one model requires approximately 8 hours.

The base architecture is a three-layer LSTM (Schmidhuber, Hochreiter, et al. 1997) with hidden dimension 1024, totalling approximately 25 M parameters. Input tokens are represented as one-hot vectors over the vocabulary, processed by the LSTM, and projected back to vocabulary size by a linear output layer. For the dropout LSTM baseline, a dropout rate of 0.2 is applied following (Segler et al. 2018). For BARNN, the inference network E_ψ consists of two fully connected layers of width 128 with LeakyReLU activation and a final sigmoid layer, producing dropout probabilities p_t , from which rates are recovered as $\alpha_t = p_t / (1 - p_t)$. After training, 50K molecules were sampled from each model to compute the reported statistics.

3.8.3 Additional Results

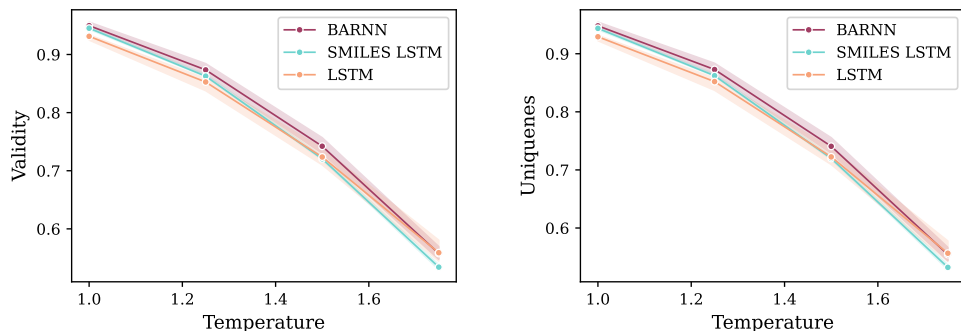


Figure 14: Validity vs. temperature. Figure 15: Uniqueness vs. temperature. BARNN maintains higher validity than BARNN retains higher uniqueness at low baselines across most temperature values. to moderate temperatures.

MOLECULAR PROPERTY DISTRIBUTIONS. We compute six molecular descriptors – MW, HBDs, HBAs, rotatable bonds, LogP, and TPSA – for both BARNN-generated molecules and the ChEMBL dataset. A t-SNE visualisation (Van der Maaten and G. Hinton 2008) of the two-dimensional descriptor space (Figure 12) shows substantial overlap between the two distributions, indicating that BARNN captures the underlying molecular property landscape. Kernel density estimates for each descriptor (Figure 13) confirm this, with BARNN and ChEMBL distributions closely aligned even for multi-modal descriptors such as H-bond donors and acceptors.

ROBUSTNESS TO TEMPERATURE. Temperature scales the logits prior to the softmax at sampling time: lower temperatures produce more deterministic outputs, while higher temperatures increase diversity at the cost of validity. We evaluate BARNN and the baselines across a range of temperatures, examining validity and uniqueness (Figure 15). All models show declining validity and uniqueness with increasing temperature. At lower temperatures, BARNN maintains higher validity and uniqueness than the baselines; at very high temperatures, all models converge to comparable performance.

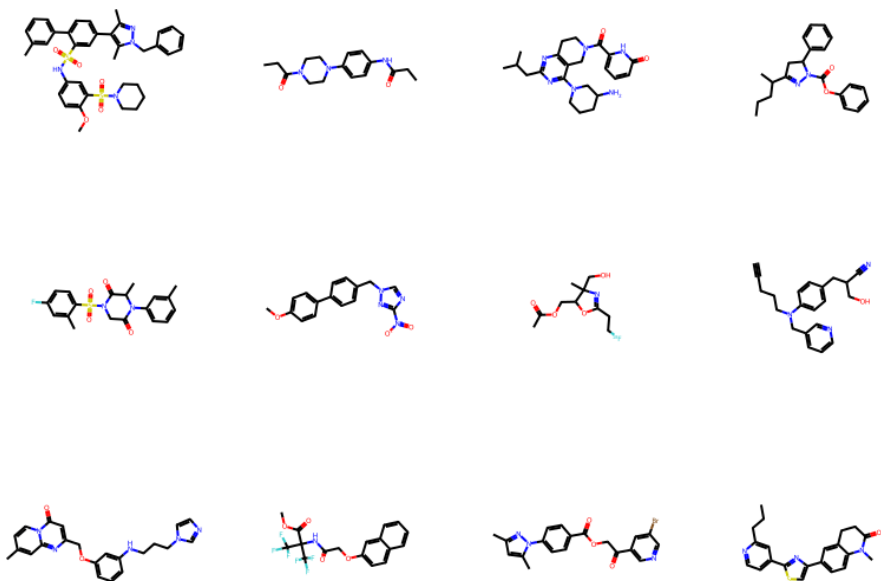


Figure 16: Generated molecules. Representative samples produced by BARNN.

3.9 ALGORITHMS AND SOFTWARE

3.9.1 Pseudocodes

Algorithm 1 Local reparameterisation trick. Computes the output of a linear layer using the local reparameterisation trick. Requires the dropout coefficient α^l for layer l and the static weights θ_l .

```

 $M \leftarrow \alpha^l H \theta_l$  {Compute the mean}
 $V \leftarrow (\alpha^l H)^2 \theta_l^2$  {Compute the variance (elementwise power)}
 $E \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$  {Sample Gaussian noise}
return  $M + \sqrt{V} \odot E$ 

```

Algorithm 2 Training Bayesian Neural PDE Solvers. At each iteration, a random timestep is drawn, dropout rates are computed by the inference network, a forward pass is performed via the local reparameterisation trick, and the loss combines the supervised MSE term with the KL regularisation. $\{\mathbf{y}_0^k, \dots, \mathbf{y}_T^k\}_{k=1}^N$: data trajectories; **NO**: neural operator with static weights θ ; E_ψ : variational posterior inference network.

```

while not converged do
   $t \leftarrow \mathcal{U}[1, T]$  {Draw a random timestep}
   $\alpha_t \leftarrow E_\psi(\mathbf{y}_{t-1}^k)$  {Compute per-layer dropout rates}
   $\hat{\mathbf{y}}_t^k \leftarrow \text{NO}(\mathbf{y}_{t-1}^k; \theta, \alpha_t)$  {Forward pass with local reparameterisation}
   $\mathcal{L}(\theta, \psi) \leftarrow \|\hat{\mathbf{y}}_t^k - \mathbf{y}_t\|^2 - D_{\text{KL}}(\alpha_t)$  {Loss using  $D_{\text{KL}}$  from Equation (43)}
  Optimise  $\theta, \psi$  by gradient descent on  $\mathcal{L}$ 
end while

```

Algorithm 3 Training Bayesian Recurrent Neural Networks. At each iteration, dropout rates are computed from the current input and hidden state, a forward pass is performed via the local reparameterisation trick, and the loss combines cross-entropy with KL regularisation. $\{\mathbf{y}_0^k, \dots, \mathbf{y}_T^k\}_{k=1}^N$: tokenised sequences; **RNN**: recurrent architecture with static weights θ ; $\mathbf{o}_t$: output probabilities at step t ; E_ψ : variational posterior inference network.

```

while not converged do
   $\mathbf{h} \leftarrow \mathbf{0}$  {Initialise hidden state}
  for  $t \in [0, \dots, T-1]$  do
     $\alpha_t^y \leftarrow E_\psi(\mathbf{y}_t)$  {Dropout rates from current input}
     $\alpha_t^h \leftarrow E_\psi(\mathbf{h})$  {Dropout rates from hidden state}
     $\mathbf{o}_{t+1}, \mathbf{h} \leftarrow \text{RNN}(\mathbf{y}_t, \mathbf{h}; \theta, \alpha_t^y, \alpha_t^h)$  {Forward pass with local reparameterisation}
  end for
   $\mathcal{L}(\theta, \psi) \leftarrow \text{CrossEntropyLoss}(\mathbf{o}_{1:T}, \mathbf{y}_{1:T}) - D_{\text{KL}}(\alpha_t)$ 
  Optimise  $\theta, \psi$  by gradient descent on  $\mathcal{L}$ 
end while

```

3.9.2 *Software*

We perform the PDEs experiment using PINA (Coscia, Ivagnes, et al. 2023) software, which is a Python library based on PyTorch (Paszke et al. 2019) and PyTorch Lightning (Falcon and The PyTorch Lightning team 2019) used for Scientific Machine Learning and includes Neural PDE Solvers, Physics Informed Networks and more. For the Molecules experiments we used PyTorch Lightning (Falcon and The PyTorch Lightning team 2019) and RDKit for postprocessing analysis of the molecules.

BAYESIAN LEARNED INTERATOMIC POTENTIALS

Machine Learning Interatomic Potentials have become a central tool in simulation-based chemistry, enabling atomistic simulations at a fraction of the cost of quantum-mechanical calculations. Despite their strong predictive capabilities, MLIPs lack a principled framework for Uncertainty Quantification, which is key in scientific applications such as active learning and the validation of simulation results against quantum references. We propose Bayesian Learned Interatomic Potential (BLIP), a scalable, architecture-agnostic variational Bayesian framework for training or fine-tuning MLIPs, built on an adaptive extension of Variational Dropout. BLIP integrates seamlessly with equivariant message-passing architectures and delivers well-calibrated uncertainty estimates with minimal computational overhead at inference time. Empirical results on simulation-based computational chemistry tasks demonstrate that BLIP improves predictive accuracy over standard MLIPs and provides trustworthy uncertainty estimates, particularly in data-scarce and out-of-distribution regimes. Fine-tuning pretrained MLIPs with BLIP further yields consistent performance gains alongside calibrated uncertainties.

4.1 INTRODUCTION

Message Passing Neural Networks (MPNNs) (Scarselli et al. 2008; Kipf and Welling 2017; Gilmer et al. 2017) have become the dominant architecture for atomistic machine learning, owing to their ability to encode physical symmetries natively such as translational, rotational, and permutation invariance (T. Cohen and Welling 2016; Weiler et al. 2023). Building on this technology, Machine Learning Interatomic Potentials (MLIPs) (Batatia et al. 2022; Wood

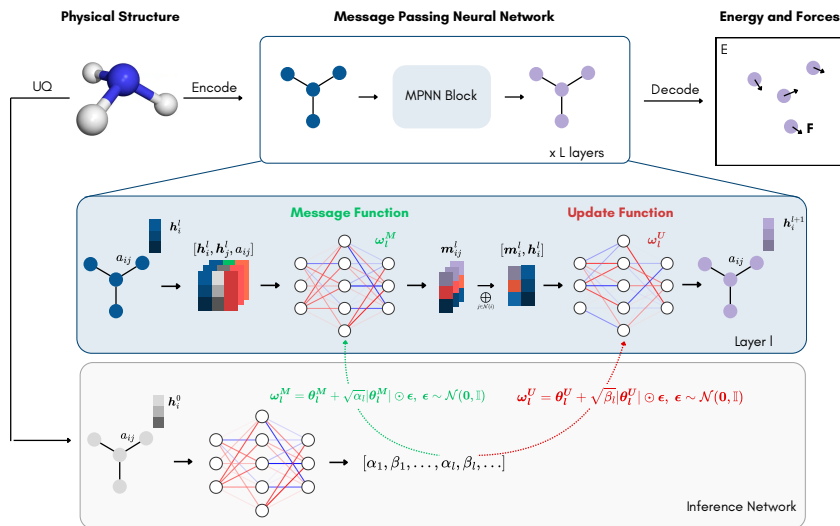


Figure 17: Bayesian Learned Interatomic Potentials (BLIP). BLIP introduces input-dependent stochastic perturbations to the weights of any message passing neural network, turning a deterministic MLIP into a principled probabilistic model trained via variational inference.

et al. 2025; Neumann et al. 2024) have emerged as efficient surrogates for quantum-mechanical simulations, enabling large-scale atomistic modelling at near *ab initio* accuracy and dramatically accelerating research in materials discovery, catalysis, and molecular dynamics.

Despite these advances, MLIPs share a fundamental limitation with most deep learning models: they are prone to silent failure on out-of-distribution inputs, and they produce no uncertainty estimates by construction. In simulation-based chemistry this is particularly consequential. Training data typically comes from Density Functional Theory (DFT) calculations, which are expensive and scale poorly with system size, making distributional shift at test time unavoidable (Tan et al. 2023). Without reliable uncertainty estimates, there is no mechanism to detect when a simulation has left the reliable region of the model – a critical gap both for validating simulation results and for driving active learning pipelines that decide where to query new DFT calculations.

This chapter addresses that gap. We introduce Bayesian Learned Interatomic Potentials (BLIPs), a variational Bayesian framework that converts any MLIP into a probabilistic model with well-calibrated uncertainty estimates and minimal computational overhead (see Figure 17). BLIP builds on an adaptive extension of Variational Dropout (Durk P Kingma et al. 2015), perturbing the MPNN weights stochastically as a function of the input, and integrates seamlessly with both invariant and equivariant architectures. We show that BLIP improves predictive accuracy over deterministic baselines, provides trustworthy uncertainty estimates in data-scarce and out-of-distribution regimes, and that fine-tuning pretrained MLIPs with BLIP consistently outperforms both single-model and ensemble alternatives.

4.2 BACKGROUND AND RELATED WORK

In this section, we present the relevant background on Message Passing Neural Networks and Machine Learning Interatomic Potentials, which will later support the formulation of our proposed method.

4.2.1 Message Passing Neural Networks

Message Passing Neural Networks (Scarselli et al. 2008; Kipf and Welling 2017; Gilmer et al. 2017) are a class of (permutation equivariant) Neural Networks that process graph-structured data. Given a Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$, and edges $e_{ij} \in \mathcal{E}$, the MPNNs operations for each layer $l = 0, \dots, L - 1$ can be summarised as:

$$\begin{aligned} \text{message+aggregate: } m_i^{l+1} &= \bigoplus_{j \in \mathfrak{N}(i)} M_l(h_j^l, a_{ij}) \\ \text{update: } h_i^{l+1} &= U_l(h_i^l, m_i^{l+1}) \end{aligned} \tag{67}$$

where h_i^l is the feature vector for node i at layer l , a_{ij} are the edge attributes, and $\bigoplus_{j \in \mathfrak{N}(i)}$ is a permutation invariant pooling operation over the node i neighbors $\mathfrak{N}(i)$. Finally, M_l, U_l are learnable message and update functions usually approximated with small Neural Networks.

4.2.2 *Machine Learning Interatomic Potentials*

Density Functional Theory (DFT) (Hohenberg and Kohn 1964; Kohn and Sham 1965) is a widely used quantum mechanical method for calculating a range of properties for various atomic systems. Under the Born-Oppenheimer approximation (Born and Heisenberg 1985), the Potential Energy Surface (PES) $E(\mathbf{r}, \mathbf{Z})$ of an atomic system is fully determined by the nuclear positions $\mathbf{r}$ and atomic numbers $\mathbf{Z}$, with the interatomic forces recovered as its negative gradient, $\mathbf{F} = -\nabla_{\mathbf{r}}E(\mathbf{r}, \mathbf{Z})$. However, DFT calculations are computationally expensive due to their cubic scaling with respect to the number of electrons; therefore, Machine Learning Interatomic Potentials (MLIPs) based on (equivariant) MPNNs (Batatia et al. 2022; X. Fu et al. 2025; Neumann et al. 2024; Rhodes et al. 2025; Schütt, Kindermans, et al. 2017) have emerged as efficient surrogates showing remarkable success, particularly due to linear scaling with system size and ability to generalise across chemical domains (Wood et al. 2025). Despite these advances, MLIPs often fail to generalise to out-of-distribution data, such as out-of-equilibrium structures or rare chemical compositions, highlighting the need for principled UQ.

4.2.3 *Uncertainty Quantification in Machine Learning Interatomic Potentials*

Uncertainty Quantification in MLIPs plays a central role in ensuring model reliability (Thaler et al. 2023; Tan et al. 2023; Chmiela et al. 2019; Kreiman and Krishnapriyan 2025), as well as in guiding efficient exploration of chemical space through active learning (Kulichenko, Nebgen, et al. 2024; Jinnouchi et al. 2020; Zaverkin et al. 2024). Among existing approaches, Deep Ensembles MLIPs (Tan et al. 2023; Oord et al. 2023; Kulichenko, Barros, et al. 2023) remain one of the most effective UQ methods in terms of both predictive accuracy and uncertainty estimation. However, Deep Ensembles are computationally expensive, with training and inference costs scaling linearly with ensemble size (one model needs to be trained/queried for each ensemble member). Moreover, they are known to exhibit calibration issues (Pernot 2022), often requiring post-hoc calibration techniques (Y. Hu et al. 2022). Finally, for large-scale foundation models, constructing and maintaining multiple ensemble members becomes typically infeasible due to substantial computational and storage demands,

leaving only a single pretrained model available for downstream tasks. For these reasons, we present a Bayesian approach that is scalable, requiring only a single trained model, and empirically exhibits good calibration, offering a practical and efficient solution for UQ in MLIPs.

4.3 METHODS

BLIP provides a practical and easy-to-use framework for training or fine-tuning any MLIP through variational Bayesian inference. The weights of the message and update functions are treated stochastically by injecting input-dependent Gaussian noise, enabling the model to capture uncertainty that varies with the atomic input structure. In what follows, we describe how BLIP integrates seamlessly into standard MLIP architectures, how it is trained across diverse tasks, and how it scales to support large models.

4.3.1 Building Bayesian Interatomic Potentials

As discussed above, MLIPs are typically implemented via MPNNs, where both the message and update functions are parameterised by dedicated neural networks. Following the Bayesian paradigm, we treat the message and update functions M_l and U_l at each layer l as Bayesian neural networks. Concretely, these functions are parameterised by random weights ω_l^M and ω_l^U , drawn independently from prior distributions $p(\omega_l^M, \omega_l^U) = p(\omega_l^M)p(\omega_l^U)$. This formulation allows uncertainty in the parameters of both functions to propagate across layers through the message passing procedure.

Under a complete factorisation across layers, the generative model for the hidden state $\mathbf{h}_i^{l+1}$ at layer $l+1$ reads:

$$p(\mathbf{h}_i^{l+1}, \mathbf{h}_i^l, \omega_l^M, \omega_l^U) = p(\mathbf{h}_i^{l+1} | \mathbf{h}_i^l, \omega_l^M, \omega_l^U) p(\omega_l^M) p(\omega_l^U). \quad (68)$$

where the conditional distribution is the pushforward induced by the standard message passing procedure:

$$\begin{aligned} \mathbf{m}_i^{l+1} &= \bigoplus_{j \in \mathfrak{N}(i)} M_l(\mathbf{h}_j^l, a_{ij}; \omega_l^M) \\ \mathbf{h}_i^{l+1} &= U_l(\mathbf{h}_i^l, \mathbf{m}_i^{l+1}; \omega_l^U) \end{aligned} \quad (69)$$

which corresponds to a deterministic delta distribution:

$$p(\mathbf{h}_i^{l+1} | \mathbf{h}_i^l, \omega_l^M, \omega_l^U) = \delta(\mathbf{h}_i^{l+1} - g(\mathbf{h}_i^l; \omega_l^M, \omega_l^U)), \quad (70)$$

with g the composition of the message and update functions defined above. At a high level, the model operates by first sampling random weights for the message and update functions at each layer, and then executing the usual message-passing updates from the initial hidden state using those sampled weights. The final output $\mathbf{y}$ is obtained from the last layer’s hidden state $\mathbf{h}^L$, which depends on the initial node features, edge attributes, and all sampled weights up to that layer: $\mathbf{h}^L = f(\mathbf{h}^0, a_{ij}, \{\omega_l^M, \omega_l^U\}_{l=0}^{L-1})$.

4.3.2 Variational Adaptive Dropout

VARIATIONAL POSTERIOR: To train the model, we define a variational posterior over the weights conditioned on the input features (the initial node representations $\mathbf{h}^0$ and edge attributes a_{ij}), so that the model can account for input-dependent uncertainty. The posterior factorises as:

$$q_\phi(\omega_l^M, \omega_l^U | \mathbf{h}^0, a_{ij}) = q_\phi(\omega_l^M | \mathbf{h}^0, a_{ij}) q_\phi(\omega_l^U | \mathbf{h}^0), \quad (71)$$

where ϕ collects the parameters of the approximate distribution. Following Coscia, Welling, et al. (2025), we adopt an efficient parameterisation based on Variational Dropout (Durk P Kingma et al. 2015), which enables scalable inference in large MPNN architectures via the local-reparametrisation trick (Durk P Kingma et al. 2015) (see Algorithm 4). We call the approach Variational Adaptive Dropout (VAD). Within each message-passing layer l , the message and update functions are realised as multi-layer perceptrons (MLPs) consisting of successive linear layers. For each internal linear layer $s = 1, \dots, S$, the weights are modelled as Gaussian distributions¹:

$$\begin{aligned} q_\phi(\omega_{ls}^M | \mathbf{h}^0, a_{ij}) &= \mathcal{N}(\theta_{ls}^M, \alpha_l(\theta_{ls}^M)^2), \\ q_\phi(\omega_{ls}^U | \mathbf{h}^0) &= \mathcal{N}(\theta_{ls}^U, \beta_l(\theta_{ls}^U)^2), \end{aligned} \quad (72)$$

¹ We assume the same number of internal layers for both message and update networks for notational simplicity, though the approach trivially generalises. We also factorise across this dimension.

where θ_{ls}^M and θ_{ls}^U are the mean weights of the message and update networks respectively, with squaring applied element-wise. The VAD coefficients α_l and β_l are input-dependent:

$$\alpha_l = \alpha_l(\mathbf{h}^0, a_{ij}) \in \mathbb{R}^+, \quad \beta_l = \beta_l(\mathbf{h}^0) \in \mathbb{R}^+, \quad (73)$$

computed per edge and node respectively by first producing a value $r \in [0, 1]$ and applying the transformation $\frac{r}{1-r}$. Equivalently, this can be viewed as injecting Gaussian noise into the weights, scaled by the corresponding variational dropout coefficients:

$$\begin{aligned} \omega_{ls}^M &= \theta_{ls}^M + \epsilon_{ls}^M \odot |\theta_{ls}^M|, & \epsilon_{ls}^M &\sim \mathcal{N}(0, \alpha_l(\mathbf{h}^0, a_{ij})), \\ \omega_{ls}^U &= \theta_{ls}^U + \epsilon_{ls}^U \odot |\theta_{ls}^U|, & \epsilon_{ls}^U &\sim \mathcal{N}(0, \beta_l(\mathbf{h}^0)), \end{aligned} \quad (74)$$

where $\odot$ denotes element-wise multiplication. The Gaussian assumption makes the local reparameterisation trick (Durk P Kingma et al. 2015) applicable, so that activations can be sampled directly rather than weights. This reduces the number of required random samples per iteration to one per neuron activation rather than one per weight, yielding lower-variance gradient estimates and improved computational efficiency (see Appendix 4.9).

ENCODING VARIATIONAL ADAPTIVE DROPOUT COEFFICIENTS The adaptive variance terms α_l and β_l are produced by a dedicated inference network E_ψ with two separate output heads, shared across all layers l . The coefficients are therefore computed once from the initial node features $\mathbf{h}^0$ and edge attributes a_{ij} , and broadcast to every message-passing layer. Consequently, the full set of trainable variational parameters ϕ comprises: (i) the unperturbed weights of the message-passing network, $\Theta = \{\theta_l^M, \theta_l^U\}_{l=0}^{L-1}$; and (ii) the inference network parameters ψ , which control the input-dependent noise injected at each layer.

EVIDENCE LOWER BOUND OPTIMISATION The variational parameters ϕ are optimised by maximising the Evidence Lower Bound (ELBO), which

balances data fit through the expected log-likelihood against model complexity measured by the KL divergence between posterior and prior:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}) = & \mathbb{E}_{q_{\boldsymbol{\phi}}} \left[\log p(\mathbf{y} \mid \mathbf{h}^0, a_{ij}, \{\boldsymbol{\omega}_l^M, \boldsymbol{\omega}_l^U\}_{l=0}^{L-1}) \right] \\ & - \lambda \sum_{l=0}^{L-1} \left(D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_l^M \mid \mathbf{h}^0, a_{ij}) \parallel p(\boldsymbol{\omega}_l^M)] \right. \\ & \left. + D_{\text{KL}}[q_{\boldsymbol{\phi}}(\boldsymbol{\omega}_l^U \mid \mathbf{h}^0) \parallel p(\boldsymbol{\omega}_l^U)] \right). \end{aligned} \quad (75)$$

where the expectation is taken over all stochastic weights $\{\boldsymbol{\omega}_l^M, \boldsymbol{\omega}_l^U\}_{l=0}^{L-1}$, and λ is the KL weighting, as commonly adopted in variational inference (Higgins et al. 2017). In practice, the expectation is approximated via a single Monte Carlo sample from the variational posterior at each training step.

We adopt a zero-mean Gaussian prior with fixed layerwise variance, matched to the variance induced by dropout with probability $p \in [0, 1]$:

$$p(\boldsymbol{\omega}_{ls}^M) = \mathcal{N}\left(\mathbf{0}, \frac{p}{1-p}(\boldsymbol{\theta}_{ls}^M)^2\right), \quad p(\boldsymbol{\omega}_{ls}^U) = \mathcal{N}\left(\mathbf{0}, \frac{p}{1-p}(\boldsymbol{\theta}_{ls}^U)^2\right), \quad (76)$$

for each message-passing layer l and internal linear layer s . This choice yields closed-form KL divergences for both terms in (75), as derived in Appendix 4.6.1. Algorithm 5 summarises the full training procedure for MLIPs using BLIP.

4.3.3 Equivariance and Invariance Analysis

Many MLIP architectures are explicitly designed to be either *equivariant* or *invariant* to certain input transformations (T. Cohen and Welling 2016; Weiler et al. 2023; Bronstein et al. 2017; T. S. Cohen et al. 2019). An *equivariant* network guarantees that transforming the input (e.g., via translation or rotation) produces a corresponding transformation in the output, while an *invariant* network is a special case where the output remains unchanged under such transformations. Both properties are essential when modelling atomic systems, since geometric symmetries should leave the model’s predictions unaffected. BLIP is compatible with *equivariant* and *invariant* MLIPs without compromising their symmetry properties². To guarantee this, the variational dropout

² Although the model is equivariant in distribution, individual forward passes may break exact equivariance or invariance due to the injected noise. For a given input, different noise samples

coefficients α and β must themselves be invariant, which is achieved by implementing the inference network E_ψ as an invariant neural network. When α and β are invariant to input transformations, the local reparameterisation trick can be safely applied to any equivariant network whose learnable layers can be expressed as unconstrained linear transformations without an irreducible representation (irrep) dimension. This includes architectures such as those in (Satorras et al. 2021; Schütt, O. Unke, et al. 2021; Köhler et al. 2019; Schütt, Kindermans, et al. 2017). In its current form, however, it excludes other important families of equivariant MLIPs, such as MACE (Batatia et al. 2022) and UMA (Wood et al. 2025), motivating future work to close this gap. A more detailed discussion is provided in Appendix 4.6.3.

4.4 EXPERIMENTS

We demonstrate the effectiveness of BLIP across a range of simulation-based chemistry tasks. As an initial illustrative example, BLIP is applied to model a system of charged particles interacting via Coulomb forces. Subsequently, we consider three challenging MLIP benchmarks: (i) training from scratch using the equivariant PaiNN architecture (Schütt, O. Unke, et al. 2021) on a data-scarce, out-of-distribution ammonia (NH_3) system; (ii) fine-tuning the pretrained ORB v3 model (Rhodes et al. 2025) on the silica glass (SiO_2) system with 699 atoms; and (iii) fine-tuning the same pretrained model on the MATPES (A. D. Kaplan et al. 2025) PBE dataset using active learning.

4.4.1 Charged Particle System

The Coulomb N-body particle benchmark (Fuchs et al. 2020) is used as a first test case to assess BLIP’s uncertainty quantification capabilities. The system consists of 5 charged particles (positive or negative), each characterised by a position and velocity in 3D space. The objective is to predict particle positions after 1000 timesteps given their initial conditions. We follow the same training and test split as Satorras et al. (2021), and train all models using the standard

(which themselves need not be equivariant) can yield different outputs. However, averaging across samples preserves the equivariance of the statistical moments.

Table 10: Results for the N-body system. Averages are computed over 4 random initialisation seeds.

Equivariant	Model	MSE $\times 10^{-1}$ ($\downarrow$)	NLL ($\downarrow$)	CRPS ($\downarrow$)
$\times$	GNN (base model)	$0.121^{\pm 0.004}$	-	-
	MC Dropout ($p = 0.2$)	$0.117^{\pm 0.001}$	$-7.83^{\pm 0.31}$	$0.043^{\pm 0.000}$
	MC Dropout ($p = 0.5$)	$0.138^{\pm 0.049}$	$-8.03^{\pm 2.13}$	$0.046^{\pm 0.011}$
	Deep Ensemble	$0.106^{\pm \text{NA}}$	$-3.89^{\pm \text{NA}}$	$0.034^{\pm \text{NA}}$
	BLIP (ours)	$0.092^{\pm 0.004}$	$-9.03^{\pm 0.50}$	$0.032^{\pm 0.001}$
$\checkmark$	EGNN (base model)	$0.069^{\pm 0.002}$	-	-
	MC Dropout ($p = 0.2$)	$0.099^{\pm 0.006}$	$-6.19^{\pm 0.42}$	$0.044^{\pm 0.006}$
	MC Dropout ($p = 0.5$)	$0.299^{\pm 0.011}$	$0.91^{\pm 0.26}$	$0.091^{\pm 0.001}$
	Deep Ensemble	$0.057^{\pm \text{NA}}$	$-16.22^{\pm \text{NA}}$	$0.020^{\pm \text{NA}}$
	BLIP (ours)	$0.060^{\pm 0.003}$	$-12.52^{\pm 0.72}$	$0.022^{\pm 0.001}$

mean squared error loss, augmented with a KL divergence term for BLIP. Further dataset and metric details are provided in Section 4.7.

Two architectures are benchmarked: a standard Graph Neural Network (GNN) (Gilmer et al. 2017) and the Equivariant Graph Neural Network (EGNN) of Satorras et al. (2021), both using the same network hyperparameters and training protocol as in (Satorras et al. 2021). For uncertainty quantification, we compare against two Bayesian baselines built on the same architectures: Monte Carlo (MC) Dropout (Gal and Ghahramani 2016) at different dropout probabilities p , and Deep Ensembles (Lakshminarayanan et al. 2017) with four ensemble members. For BLIP, a compact invariant inference network is employed for both the message and update function coefficients. Additional architecture and training details are given in Section 4.8.

RESULTS: Table 10 reports predictive performance and uncertainty quantification metrics on the particle system, averaged over four random seeds. BLIP consistently outperforms MC Dropout across all tested dropout probabilities, and matches or exceeds Deep Ensembles in both accuracy and uncertainty quality. In the non-equivariant setting (GNN), BLIP attains the lowest MSE and best-calibrated uncertainty estimates, as reflected by the lowest NLL and CRPS.

In the equivariant setting (EGNN), Deep Ensembles achieve the best overall performance, though only marginally ahead of BLIP. Crucially, BLIP remains highly competitive while being substantially more efficient, relying on a single model for both training and inference rather than the four models required by Deep Ensembles ($\times 4$ parameter cost). Taken together, these results confirm that BLIP delivers high-quality uncertainty estimates without sacrificing predictive accuracy, and at a lower computational cost than ensemble-based alternatives.

4.4.2 *Learning Potentials in Data-Scarce and Out-Of-Distribution Regimes*

This experiment examines the ammonia (NH_3) molecule as a representative benchmark for assessing robustness in a realistic low-data, high out-of-distribution (OOD) regime — a common challenge in materials science, where training data are often biased towards equilibrium or near-equilibrium configurations. We use the dataset introduced by Tan et al. (2023), comprising a small collection of NH_3 molecular geometries sampled across a range of energy configurations. The training set contains 78 near-equilibrium geometries with energies between 0 and 5 kcal/mol, while the test set consists of 129 OOD geometries with energies up to 70 kcal/mol, thereby probing generalisation far from the training distribution. Performance is evaluated using three metrics: mean absolute error (MAE), expected calibration error (ECE), and the Spearman correlation. Further dataset and metric details are provided in Section 4.7.

The equivariant PaiNN architecture (Schütt, O. Unke, et al. 2021) is adopted as the underlying MLIP. To ensure a fair comparison, we follow the same hyperparameter settings, training protocol, and evaluation procedure as in Tan et al. (2023). Energy conservation is explicitly enforced during training by computing atomic forces as the negative gradient of the predicted potential energy with respect to atomic positions, with energy and force terms weighted as in Tan et al. (2023). As before, BLIP uses a compact invariant inference network for the message and update function coefficients, with full details given in Section 4.8.

Table 11: Energy and force MAE on OOD ammonia. Mean absolute errors for energy and forces, averaged over four random seeds. Baseline values are from (Tan et al. 2023).

Model	MAE ($\downarrow$)	
	Energy [kcal/mol]	Forces [kcal/mol/Å]
Deep Ensemble	11.71	48.03
Evidential Regression	13.58	53.29
Mean Variance Estimate	13.47	53.56
Gaussian Mixture Model	12.51	50.17
BLIP (ours)	11.51 ± 0.75	41.93 ± 3.38

RESULTS: As shown in Table 11, BLIP achieves the best MAE across both energy and force targets, surpassing all baseline models reported in (Tan et al. 2023). For energy prediction, BLIP obtains an MAE of 11.51 ± 0.75 , marginally better than the Deep Ensemble (11.71) and substantially lower than the remaining probabilistic baselines. On force prediction, a more demanding task, the improvement is more pronounced: BLIP reaches an MAE of 41.93 ± 3.38 , well below both the Deep Ensemble and other competing methods. These results indicate that incorporating BLIP during training yields superior predictive accuracy on both targets, without any modification to the underlying architecture or the need for multiple models as required by Deep Ensembles. Turning to uncertainty estimation, Table 12 shows that BLIP achieves the lowest ECE among all models, outperforming even the Deep Ensemble. Furthermore, the Spearman correlation analysis between predicted uncertainty and squared error (Figure 18) reveals a strong linear relationship (0.863 ± 0.024), confirming that BLIP reliably identifies structures where MLIP predictions are likely to be inaccurate — a property of direct relevance to active learning.

4.4.3 Fine-Tuning Pretrained Interatomic Potentials on SiO_2

This experiment considers fine-tuning a pretrained MLIP on SiO_2 . Given the increasing scale of modern models, training universal potentials from scratch is often infeasible without substantial computational resources, making fine-tuning of existing pretrained models the practical standard. Among recent developments, we select the ORB v3 Direct 2oMat model (Rhodes et al. 2025), a state-of-the-art direct force field that is not rotationally equivariant. Fine-tuning is performed on the silica glass (SiO_2) dataset from Tan et al. (2023), which comprises system snapshots extracted from diverse long molecular dynamics trajectories, each containing 699 atoms (233 silicon and 466 oxygen). This benchmark is both realistic and challenging, as the large bulk structures make DFT calculations computationally prohibitive. Models are fine-tuned using varying numbers of training samples (32, 128, and 1024), and evaluated on a held-out test set of 373 samples using MAE, ECE, and Spearman correlation. Further details on the dataset and evaluation metrics are given in Section 4.7.

The non-conservative ORB v3 model, originally pretrained on a subset of ab-initio molecular dynamics data from OMAT 24 (Barroso-Luque et al. 2024), is fine-tuned both with (**w/**) and without (**w/o**) BLIP. A Deep Ensemble of 4 members is also fine-tuned to serve as a direct UQ comparison. Additional architectural and training details are provided in Section 4.8.

RESULTS: Figure 19 presents the fine-tuning results on the Silica dataset. For reference, the caption includes the best-performing model reported in the literature on this benchmark — a Deep Ensemble Equivariant PaiNN (4 members) from Tan et al. (2023) — which achieves a substantially higher error than even the pretrained ORB v3 baseline (**pretrained**), underscoring the advantage of universal MLIPs. For inference with the BLIP fine-tuned models, two strategies are evaluated: (i) the sample mean from 100 stochastic forward passes (**sample mean**, Algorithm 6), and (ii) a single forward pass with the unperturbed weights, i.e., the maximum a posteriori estimate (**MAP estimate**, Algorithm 7).

Both inference strategies show that BLIP is accurate and computationally efficient. Whether using the sample mean or the more lightweight MAP

Table 13: Energy and force MAE on MATPES PBE. Means and standard deviations over three random seeds.

Model	MAE ($\downarrow$)	
	Energy [meV/atom]	Forces [meV/Å]
w/o BLIP	111.83 $\pm$ 1.24	110.84 $\pm$ 0.57
w/ BLIP + random	96.65 $\pm$ 0.09	105.12 $\pm$ 0.06
w/ BLIP + uncertainties	91.13$\pm$1.14	103.85$\pm$0.14

estimate, BLIP outperforms all ORB v3 baselines as well as the pretrained PaiNN ensemble. Notably, MAP-based inference with BLIP surpasses the deep-ensembled fine-tuned models at a fraction of the computational cost, since it requires storing and fine-tuning only a single model. Regarding uncertainty quantification on the 1024-sample setup, BLIP achieves an ECE of 0.032 ± 0.002 , compared to 0.247 for the Deep Ensemble. For the Spearman correlation between predicted variance and squared error — a key indicator for active learning — BLIP obtains 0.602 ± 0.002 versus 0.676 for the Deep Ensemble, representing a slight underperformance on this metric. Overall, these findings confirm that fine-tuning with BLIP yields superior predictive accuracy over both deterministic and ensemble baselines, with well-calibrated uncertainties, while preserving the same inference cost as a standard deterministic model.

4.4.4 Active Learning Fine-Tuning of Pretrained Interatomic Potentials

In this experiment, the ORB v3 Direct 20OMat model is fine-tuned on the MATPES PBE dataset (A. D. Kaplan et al. 2025), which comprises roughly 400,000 carefully curated structures drawn from 281 million molecular dynamics snapshots, covering 16 billion atomic environments.

To improve model accuracy efficiently, an Active Learning (AL) strategy is employed, where BLIP uncertainties guide the selection of structures for which the pretrained model is most uncertain. These high-uncertainty structures are computed at the reference DFT level and used to iteratively fine-tune the

model. By repeating this cycle, the model progressively incorporates the most informative data points, maximising performance gains while minimising computational cost. Model performance is assessed via the MAE on both forces and energies, and the validation loss is tracked across AL iterations. Dataset and metric details are provided in Section 4.7, and implementation details in Section 4.8.

RESULTS: The validation loss curves in Figure 20 reveal a clear and consistent ordering across models throughout the AL iterations. Uncertainty-guided fine-tuning with BLIP (**w/ BLIP + uncertainties**), where structures are selected from a pool based on the variance in predicted energies, converges fastest and to the lowest error. Fine-tuning with BLIP on randomly selected structures (**w/ BLIP + random**) converges to a worse optimum, yet still outperforms fine-tuning without BLIP (**w/o BLIP**), which exhibits the poorest performance overall. Notably, additional AL iterations further widen the gap between uncertainty-guided and random selection. This ordering is consistently reflected in the test errors for both forces and energies reported in Table 13. Collectively, these results establish that fine-tuning with BLIP reliably improves model quality, and that exploiting BLIP uncertainties for active data selection not only accelerates convergence but also leads to systematically better predictive accuracy.

4.5 FINAL DISCUSSION

We presented BLIP, a scalable and architecture-agnostic variational Bayesian framework for principled uncertainty quantification in (equivariant) message passing neural networks, applicable to both training from scratch and fine-tuning pretrained MPNNs. In contrast to deep ensembles, BLIP preserves the inference-time efficiency of standard deterministic models for predictive tasks, while providing well-calibrated uncertainty estimates through stochastic sampling. The effectiveness of the proposed framework was demonstrated across a range of scientific benchmarks, spanning particle prediction and machine-learned interatomic potentials. In data-scarce and out-of-distribution regimes, BLIP achieves strong performance in both prediction and uncertainty quantification. Furthermore, fine-tuning pretrained MLIPs with BLIP consistently

improves model accuracy, outperforming deterministic and ensemble-based baselines alike. The uncertainty estimates obtained are well-calibrated and strongly correlated with model error, making them directly applicable to downstream tasks such as active learning and error detection. Taken together, these findings position BLIP as a practical drop-in tool for enhancing the predictive performance of MLIPs and MPNNs more broadly, while equipping them with reliable uncertainty estimates.

Table 12: Calibration and error correlation on OOD ammonia. Expected calibration error and Spearman correlation for forces, averaged over four random seeds. Baseline values are from (Tan et al. 2023).

Model	ECE ($\downarrow$)	ρ ($\uparrow$)
Deep Ensemble	0.108	0.825
Evidential Regression	0.120	0.779
Mean Variance Estimate	0.135	0.693
Gaussian Mixture Model	0.112	0.904
BLIP (ours)	0.091 ± 0.006	0.863 ± 0.024

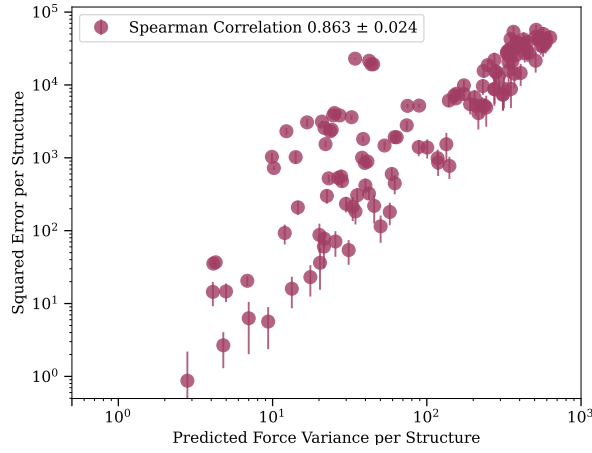


Figure 18: Correlation between error and uncertainty. Spearman correlation between predicted uncertainty and squared force error in the OOD ammonia test set.

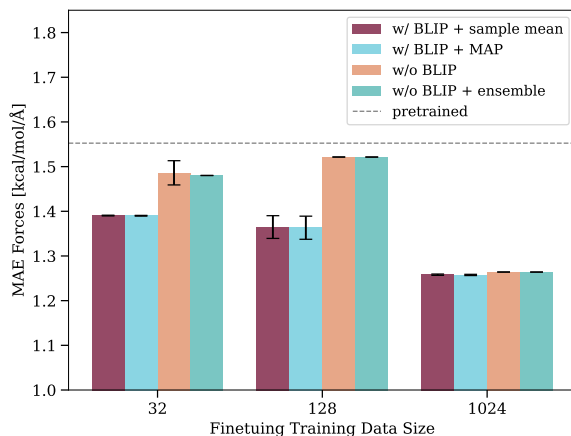


Figure 19: Forces MAE on silica across training-set sizes. Fine-tuned ORB v3 Direct 20OMAT results; error bars show the standard error over four random seeds. The best literature baseline is a four-member Deep Ensemble Equivariant PaiNN trained on 1024 configurations, with MAE 6.58 kcal/mol/Å.

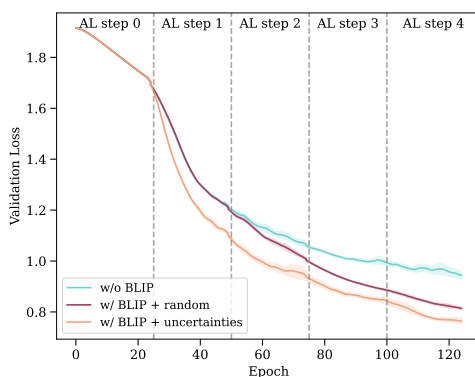


Figure 20: Active-learning validation loss. Validation loss across active-learning steps for three fine-tuning strategies, calculated as the sum of energy and force MAEs. Curves show means over three independent runs; energy is reported in eV and forces in eV/Å.

CHAPTER APPENDIX

4.6 PROOFS AND DERIVATIONS

This appendix collects the formal proofs supporting the results presented in the main text.

4.6.1 KL-Divergence Proof

We derive a closed form for the Kullback-Leiber divergence of eq. (75). Exploiting the factorisation across layers, the total KL divergence decomposes as:

$$D_{\text{KL}}^{\text{tot}} = \sum_{l=1}^L \sum_{s=1}^S \left[D_{\text{KL}}[q_{\phi}(\omega_{ls}^M \mid \mathbf{h}^0, a_{ij}) \parallel p(\omega_l^M)] + D_{\text{KL}}[q_{\phi}(\omega_{ls}^U \mid \mathbf{h}^0) \parallel p(\omega_{ls}^U)] \right]. \quad (77)$$

For the variational posterior and prior defined in the main text:

$$\begin{aligned} q_{\phi}(\omega_{ls}^M \mid \mathbf{h}^0, a_{ij}) &= \mathcal{N}\left(\theta_{ls}^M, \alpha_l(\theta_{ls}^M)^2\right), & p(\omega_{ls}^M) &= \mathcal{N}\left(0, \frac{p}{1-p}(\theta_{ls}^M)^2\right), \\ q_{\phi}(\omega_{ls}^U \mid \mathbf{h}^0) &= \mathcal{N}\left(\theta_{ls}^U, \beta_l(\theta_{ls}^U)^2\right), & p(\omega_{ls}^U) &= \mathcal{N}\left(0, \frac{p}{1-p}(\theta_{ls}^U)^2\right), \end{aligned} \quad (78)$$

each marginal divergence admits an analytic form:

$$\begin{aligned} D_{\text{KL}}[q_{\phi}(\omega_{ls}^M \mid \mathbf{h}^0, a_{ij}) \parallel p(\omega_l^M)] &= \frac{1}{2} \left(\frac{(\alpha_l + 1)(1-p)}{p} + \log\left(\frac{p}{1-p}\right) - \log(\alpha_l) - 1 \right), \\ D_{\text{KL}}[q_{\phi}(\omega_{ls}^U \mid \mathbf{h}^0, a_{ij}) \parallel p(\omega_l^U)] &= \frac{1}{2} \left(\frac{(\beta_l + 1)(1-p)}{p} + \log\left(\frac{p}{1-p}\right) - \log(\beta_l) - 1 \right). \end{aligned} \quad (79)$$

In practice, the KL term is weighted by a factor λ , as is standard in ELBO optimisation (Alemi et al. 2018; Higgins et al. 2017).

4.6.2 Predictive Distribution and Uncertainty Estimates

This section describes how BLIP is used for inference and uncertainty quantification. During inference, the goal is to regress the output from the input by marginalising over the weights; for UQ, the aim is to decompose epistemic (weight) and aleatoric (data) uncertainty.

PREDICTIVE DISTRIBUTION. The model *predictive distribution* is obtained by marginalising over the weights ω drawn from the posterior:

$$p(\mathbf{y} \mid \mathbf{h}^L) = \mathbb{E}_{q_\phi} \left[p \left(\mathbf{y} \mid \mathbf{h}^L = f(\mathbf{h}^0, a_{ij}, \{\omega_l^M, \omega_l^U\}) \right) \right]. \quad (80)$$

In practice, this is approximated by performing multiple stochastic forward passes and averaging the outputs, following the standard Monte Carlo approach to predictive distributions (Gal and Ghahramani 2016). When uncertainty is not required, a deterministic approximation is obtained by evaluating the network at the posterior mean weights.

UNCERTAINTY ESTIMATES. The predictive distribution admits a clean variance decomposition via the law of total variance:

$$\sigma^2 = \underbrace{\sigma_\omega^2(\mathbb{E}[\mathbf{y} \mid \mathbf{h}^L])}_{\text{epistemic uncertainty}} + \underbrace{\mathbb{E}_\omega[\sigma^2(\mathbf{y} \mid \mathbf{h}^L)]}_{\text{aleatoric uncertainty}}, \quad (81)$$

where $\mathbb{E}[\mathbf{y} \mid \mathbf{h}^L]$ and $\sigma^2(\mathbf{y} \mid \mathbf{h}^L)$ denote the mean and variance of $p(\mathbf{y} \mid \mathbf{h}^L)$ respectively.

4.6.3 BLIP Stochastic Equivariance and Local Reparametrisation

This section presents the formal treatment of equivariance in BLIP. We begin with a general proposition for stochastic equivariant maps, and subsequently connect it to the BLIP update rule.

STOCHASTIC EQUIVARIANT MAPS. Let G be a group acting measurably on spaces X and Y , with $g \cdot x \in X$ and $g \cdot y \in Y$ denoting the respective actions. We assume:

1. An invariant map $\text{inv} : X \rightarrow \text{Inv}(X)$ such that

$$\text{inv}(g \cdot x) = \text{inv}(x) \quad \forall g \in G, x \in X.$$

2. A class $\mathcal{F}$ of deterministic equivariant maps $f : X \rightarrow Y$, satisfying

$$f(g \cdot x) = g \cdot f(x) \quad \forall f \in \mathcal{F}, g \in G, x \in X.$$

3. A measurable map

$$E : \text{Inv}(X) \rightarrow \mathcal{P}(\mathcal{F})$$

assigning to each invariant value a probability distribution over equivariant maps.

Define a stochastic map:

$$P[y \mid x] = \int_{\mathcal{F}} \delta(y - f(x)) dE(\text{inv}(x)), \quad (82)$$

i.e., sample $f \sim E(\text{inv}(x))$ and return the deterministic value $f(x)$.

Proposition 4.1. *The stochastic map P is G -equivariant:*

$$P[y \mid g \cdot x] = g_* P[y \mid x] \quad \forall g \in G, x \in X, \quad (83)$$

where $g_* : \mathcal{P}(Y) \rightarrow \mathcal{P}(Y)$ denotes the pushforward under the action $y \mapsto g \cdot y$.

Proof. Fix $g \in G$ and $x \in X$. By definition of P ,

$$P[y \mid g \cdot x] = \int_{\mathcal{F}} \delta(y - f(g \cdot x)) dE(\text{inv}(g \cdot x)). \quad (84)$$

Since inv is invariant, $\text{inv}(g \cdot x) = \text{inv}(x)$, giving

$$P[y \mid g \cdot x] = \int_{\mathcal{F}} \delta(y - f(g \cdot x)) dE(\text{inv}(x)). \quad (85)$$

Each $f \in \mathcal{F}$ is G -equivariant, so $f(g \cdot x) = g \cdot f(x)$. Substituting yields

$$P[y \mid g \cdot x] = \int_{\mathcal{F}} \delta(y - g \cdot f(x)) dE(\text{inv}(x)). \quad (86)$$

By the definition of the pushforward, $g_*\delta(\cdot - f(x)) = \delta(\cdot - g \cdot f(x))$, and therefore

$$P[y \mid g \cdot x] = g_* \left(\int_{\mathcal{F}} \delta(y - f(x)) dE(\text{inv}(x)) \right) = g_* P[y \mid x]. \quad (87)$$

□

Corollary 4.1. *The BLIP update rule in (74) applied to an equivariant MLIP yields an equivariant stochastic MLIP.*

The connection between Corollary 4.1 and Proposition 4.1 becomes clear through the following correspondences:

- The invariant map inv corresponds to the BLIP inference network E , which maps the input to invariant variational dropout coefficients ((73)).
- The equivariant map f is an equivariant MLIP parameterised by θ .
- The stochastic map $P[y \mid x]$ is the one induced by the BLIP update ((74)).

It is worth noting that the above corollary applies to any family of equivariant neural networks, including UMA (Wood et al. 2025) and MACE (Batatia et al. 2022). In what follows, we show that the local reparametrisation trick (sampling activations) breaks equivariance, whereas perturbing the weights directly does not, suggesting a direction for future work on equivariance-preserving activation sampling.

Algorithm 4 Local reparametrisation trick: Given a minibatch of activations h and probabilistic Gaussian weights $\omega = \theta + \alpha|\theta|\epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, compute the output of a linear layer using the local reparametrisation trick. For the first layer, h corresponds to the state input.

```

1:  $m \leftarrow \text{matmul}(h, \theta)$ 
2:  $std \leftarrow \alpha \cdot \sqrt{\text{matmul}(h \odot h, \theta \odot \theta)}$ 
3:  $eps \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
4: return  $m + std \odot eps$ 

```

LOCAL REPARAMETRISATION TRICK. The local reparametrisation trick (Durk P Kingma et al. 2015) enables efficient sampling from the predictive distribution of Bayesian linear layers. For a single linear layer with input $\mathbf{h}$ and weight matrix $\omega \sim \mathcal{N}(\boldsymbol{\theta}, \alpha \boldsymbol{\theta}^2)$, the output $\mathbf{z}$ also follows a normal distribution:

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\gamma}, \text{diag}(\boldsymbol{\delta})), \quad (88)$$

where each output element z_j satisfies

$$z_j = \gamma_j + \sqrt{\delta_j} \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, 1), \quad (89)$$

with

$$\gamma_j = \sum_i h_i \theta_{ij}, \quad \delta_j = \alpha \sum_i h_i^2 \theta_{ij}^2. \quad (90)$$

Rather than sampling weights ω directly, one samples the output $\mathbf{z}$ from a Gaussian with analytically computed mean and variance, which reduces both gradient variance and computational cost. In standard deep learning libraries, this is straightforwardly implemented by overriding the linear layer module (see Algorithm 4).

LOCAL REPARAMETRISATION FOR EQUIVARIANT LINEAR MAPS. When the local reparametrisation trick is applied, noise is transferred from the weights to the activations. If this is done on a vector with irreducible representation (irrep) dimensions, independent variances are introduced for each coordinate. Since an irrep transforms by mixing its coordinates through a fixed linear representation, all components must be scaled uniformly to preserve equivariance. The local reparametrisation trick instead assigns independent variances per channel, breaking the required coupling across the irrep. As a result, the resulting stochastic activation is no longer equivariant, even though the underlying weight distribution is.

4.7 DATASET AND METRICS

This appendix provides details on the datasets and evaluation metrics used in the experiments.

4.7.1 NBody System

DATASET GENERATION AND AVAILABILITY. We use the dataset introduced by Fuchs et al. (2020), consisting of simulations of five charged particles interacting in a three-dimensional box under Coulomb’s law. The dataset is publicly available at: https://github.com/FabianFuchsML/se3-transformer-public/tree/master/experiments/nbody/data_generation. Following Satorras et al. (2021), we generate the dataset with parameters -num-train 10000 -seed 43 -suffix small, and use 3000 samples each for training, validation, and testing with no overlap between splits.

METRICS. Three metrics are used to evaluate prediction and uncertainty estimation. Predictive accuracy is measured by the mean squared error (MSE) between predicted (mean) positions $\hat{\mathbf{y}}$ and ground-truth positions $\mathbf{y}$:

$$\text{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{3N} \sum_{i=1}^N \sum_{k=1}^3 (y_i^k - \hat{y}_i^k)^2, \quad (91)$$

where k indexes spatial dimensions and i indexes data points.

Uncertainty quality is assessed via the negative log-likelihood (NLL) and the continuous ranked probability score (CRPS). Assuming a Gaussian predictive distribution with mean $\hat{\mathbf{y}}$ and variance σ^2 , the NLL is:

$$\text{NLL}(\hat{\mathbf{y}}, \mathbf{y}, \sigma^2) = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^3 \left[\frac{(y_i^k - \hat{y}_i^k)^2}{2\sigma_i^{k2}} + \frac{1}{2} \log(2\pi\sigma_i^{k2}) \right]. \quad (92)$$

The CRPS is a proper scoring rule that compares the full predictive distribution against the observed outcome, simultaneously capturing calibration and sharpness:

$$\text{CRPS}(\hat{\mathbf{y}}, \mathbf{y}, \sigma^2) = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^3 \text{CRPS}_{\mathcal{N}}(y_i^k; \hat{y}_i^k, \sigma_i^{k2}), \quad (93)$$

where $\text{CRPS}_{\mathcal{N}}$ is the closed-form CRPS for a univariate Gaussian with mean $\hat{y}$, variance σ^2 , and observation y :

$$\text{CRPS}_{\mathcal{N}}(y; \hat{y}, \sigma^2) = \sigma \cdot \left(z(2\Phi(z) - 1) + 2\phi(z) - \frac{1}{\sqrt{\pi}} \right), \quad (94)$$

with $z = \frac{y - \hat{y}}{\sigma}$, and ϕ , Φ the standard normal PDF and CDF respectively.

4.7.2 Ammonia and Silica Glass Systems

DATASET GENERATION AND AVAILABILITY. For the ammonia system, we use the dataset of Tan et al. (2023), comprising a small collection of NH_3 molecular geometries sampled across a range of energy configurations, available at: <https://archive.materialscloud.org/records/tz6a0-hsb25>. We adopt the same training/validation/test split as Tan et al. (2023).

For the silica glass system, we likewise use the dataset from Tan et al. (2023), consisting of large SiO_2 structures extracted from diverse long molecular dynamics trajectories spanning varied physical conditions, including temperature equilibration, cooling, uniaxial tension, and compression. This dataset is available at the same URL, and we follow the same validation/test split protocol as Tan et al. (2023), varying the training set size across 32, 128, and 1024 samples. Further details on the DFT calculations used to generate both datasets are provided in the Supplementary Information of Tan et al. (2023).

METRICS. Following the experimental protocol of Tan et al. (2023), predictive accuracy is measured by the mean absolute error between predicted and true energies and forces $\hat{E}, \hat{F}$:

$$\begin{aligned}\text{MAE}(\hat{E}, E) &= \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^3 |E_i - \hat{E}_i|, \\ \text{MAE}(\hat{F}, F) &= \frac{1}{3N} \sum_{i=1}^N \sum_{k=1}^3 |F_i^k - \hat{F}_i^k|,\end{aligned}\tag{95}$$

where k indexes spatial dimensions and i indexes data points.

Uncertainty quality is evaluated via the Expected Calibration Error (ECE) on forces, following Tan et al. (2023). Forces are chosen over energies as they are more sensitive to overfitting and distributional shift, and thus provide a more informative measure of epistemic uncertainty. Per-structure uncertainty is summarised by the mean force $\bar{F}_i$, obtained by averaging per-atom forces across spatial dimensions, and the corresponding standard deviation $\bar{\sigma}_i$, computed

as the square root of the average variance norm per atom following Tan et al. (2023). The ECE is then:

$$\text{ECE} = \int_0^1 |p - p_{\text{obs}}| dp, \quad p_{\text{obs}} = \frac{1}{N} \sum_{i=1}^N \mathbb{I} \left\{ \bar{F}_i \leq Q_p[\hat{F}_i, \bar{\sigma}_i] \right\}, \quad (96)$$

where Q is the Gaussian quantile function and the integral is approximated by the trapezoidal rule. The rank correlation between predicted variance $\bar{\sigma}_i^2$ and squared error norm per atom is further assessed via the Spearman correlation coefficient, computed using the SciPy (Virtanen et al. 2020) implementation.

4.7.3 MATPES Dataset

DATASET AVAILABILITY. We use the PBE version of the MATPES dataset (A. D. Kaplan et al. 2025), publicly available at <https://matpes.ai/>. For the zero active learning (AL) step, the dataset is partitioned as follows (as fractions of the full dataset): initial training set = 0.001, pool set = 0.025, test set = 0.79, and validation set = 0.124. At each AL step, 10% of candidate structures are drawn from the pool, either at random or by selecting the top 10% most uncertain structures, where uncertainty is quantified as the standard deviation of predicted energies.

METRICS. Prediction accuracy is evaluated using the mean absolute error for energies and forces:

$$\begin{aligned} \text{MAE}(\hat{E}, E) &= \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^3 |E_i - \hat{E}_i|, \\ \text{MAE}(\hat{F}, F) &= \frac{1}{3N} \sum_{i=1}^N \sum_{k=1}^3 |F_i^k - \hat{F}_i^k|, \end{aligned} \quad (97)$$

where i indexes data points and k spatial dimensions. For uncertainty quantification, the standard deviation of predicted energies across stochastic samples is used:

$$\sigma_E = \frac{1}{N-1} \sqrt{\sum_{i=1}^N (E_i - \hat{E}_i)^2}. \quad (98)$$

Table 14: Shared training hyperparameters. Main hyperparameters across experiments. Adam refers to the optimiser of Diederik P Kingma (2014), and AdamW to that of Loshchilov and Hutter (2017).

Hyper-parameter	NBody	Ammonia	Silica	MATPES
Optimization Epochs	10000	500	50	25 (5 AL steps)
Batch Size	100	64	8	128
Optimiser	Adam	Adam	AdamW	AdamW
Loss	MSE	MAE	HuberLoss(0.01)	HuberLoss(0.01)
Maximum learning rate	$5e - 4$	$1e - 3$	$1e - 5$	$1e - 5$
Learning rate scheduling	Const.	Reduce On Plateau	Cosine	Const.
Gradient Clip Norm Value	NA	NA	0.5	0.5
Warmup Iteration Phase	NA	NA	5%	N

4.8 ARCHITECTURES AND TRAINING DETAILS

All models were trained on a single 64GB A100 Nvidia GPU. Table 14 reports the hyperparameters shared across experiments. In all cases, models are trained on the training split and evaluated on the test split. The validation split is used exclusively for checkpoint selection, retaining the model with the lowest validation loss, computed using the same loss function as during training.

4.8.1 NBody System

Two architectures are benchmarked: a standard Graph Neural Network (GNN) and the Equivariant Graph Neural Network (EGNN), as described in the main text. Both are also used as the backbone for the probabilistic baselines.

For the GNN, each particle’s initial position and velocity are embedded through a linear layer to produce the initial node features $\mathbf{h}^0$. Edge attributes are derived from pairwise charge interactions $a_{ij} = c_i c_j$, and likewise processed by a linear layer before entering message passing. Both message and update functions are four-layer MLPs of the form

$$64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow 64,$$

and a two-layer regression head

$$64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow 3$$

maps the final node features to particle positions.

For the EGNN, we use the velocity variant from Satorras et al. (2021), where the initial velocity norm is mapped through a linear layer and included in $\mathbf{h}_i^0$. Charges are encoded as edge attributes via $a_{ij} = c_i c_j$, and the message and update functions share the same MLP structure as in the GNN. Following Satorras et al. (2021), no separate regression head is used, as the network directly outputs particle positions.

For MC Dropout, we follow Gal and Ghahramani (2016) and apply dropout after all linear layers of the message-passing network at varying probabilities. For Deep Ensembles, the same architecture is trained four times with different random seeds.

For BLIP, a compact invariant posterior network with two linear layers of size 64 and swish activations is used for both the message and update coefficients. Dropout probabilities p are predicted and converted to VAD coefficients via $p/(1-p)$. The message inference head takes as input the edge attribute a_{ij} concatenated with $\mathbf{h}_i^0$ and $\mathbf{h}_j^0$, while the update inference head takes only $\mathbf{h}^0$. VAD coefficients are clamped to a maximum of 4.0 for stability. A prior dropout probability of 0.5 and a regularisation weight of $\lambda = 0.01$ are used during training.

4.8.2 Ammonia Molecule System

The PaiNN architecture (Schütt, O. Unke, et al. 2021) is adopted as the underlying MLIP. To ensure a fair comparison, we use the same hyperparameter settings as in Tan et al. (2023), available at: https://github.com/learningmatter-mit/UQ_singleNN.

For BLIP, a compact invariant posterior network is used for both the message and update coefficients:

$$64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow \text{head-dim.}$$

The message inference head takes as input the squared interatomic distance $\|\mathbf{r}_i - \mathbf{r}_j\|^2$, while the update inference head takes an embedding of the atomic

number Z . VAD coefficients are clamped to 4.0, and training uses a prior dropout probability of 0.5 with regularisation weight $\lambda = 10$.

4.8.3 *Silica Glass System*

The ORB v3 architecture (Rhodes et al. 2025), pretrained on ab-initio molecular dynamics data from OMAT 24 (Barroso-Luque et al. 2024), serves as the base MLIP. The Deep Ensemble variant fine-tunes four independent models with different initialisation seeds.

For BLIP, a deeper posterior network with four linear layers of hidden size 64 and swish activations is employed for both message and update functions:

$$64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow 64 \rightarrow \text{swish} \rightarrow \text{head-dim}.$$

The edge network takes as input the concatenation of the normalised pairwise distance vector and 20 Gaussian radial basis functions applied to the raw edge distance. The node network processes atomic number embeddings. VAD coefficients are clamped to $[10^{-5}, 0.11]$. A prior dropout probability of 0.01 and regularisation weight $\lambda = 0.001$ are used, reflecting the sensitivity of the pretrained ORB weights to perturbation, which was not part of the original training procedure. A separate learning rate of $1\text{e-}4$ is used for the inference network to allow faster adaptation relative to the pretrained backbone.

4.8.4 *MATPES Dataset*

The same hyperparameters as for the Silica Glass System are used throughout.

4.9 ALGORITHMS AND SOFTWARE

This section presents the pseudocode for training and inference with BLIP.

TRAINING WITH BLIP. Algorithm 5 outlines the training procedure. Given a graph representing an atomic configuration with initial node features h^0 and edge attributes a_{ij} , the inference network E_ψ first produces VAD dropout coefficients α, β for each Bayesian linear layer. These define the uncertainty-

aware weight distributions used in the subsequent message passing steps. At each layer $l = 0, \dots, L - 1$, node features $\mathbf{h}^l$ are updated via stochastic message passing using the local reparametrisation trick (Algorithm 4). The atomic representations $\mathbf{h}^{1:L}$ are then pooled to predict the total energy E . Forces $\mathbf{F}$ are obtained either by differentiating the energy (conservative field) or directly from $\mathbf{h}^L$ (non-conservative). The training loss combines mean absolute errors on energy and forces with the KL divergence regulariser.

Algorithm 5 Training Bayesian Learned Interatomic Potentials: Given a graph representing an atomic configuration with initial node features $\mathbf{h}^0$ and edge attributes a_{ij} , the model is trained by sampling Bayesian weights, performing stochastic message passing, and computing energies and forces. The loss combines mean absolute errors on energy and force predictions with a KL divergence regulariser depending only on the VAD coefficients.

```

1: while not converged do
2:    $\alpha, \beta \leftarrow E_\psi(\mathbf{h}^0, a_{ij})$  {Encode inputs to obtain VAD coefficients for all
    layers}
3:   for layer  $l = 0, \dots, L - 1$  do
4:      $\mathbf{h}^{l+1} \leftarrow \text{MessagePassing}(\mathbf{h}^l, a_{ij}, \alpha^l, \beta^l)$  {Stochastic message passing using Alg. 4}
5:   end for
6:   Compute energy  $E$  from  $\mathbf{h}^{1:L}$  by pooling
7:   if field is conservative then
8:      $\mathbf{F} \leftarrow -\nabla E$ 
9:   else
10:    Compute forces  $\mathbf{F}$  directly from  $\mathbf{h}^L$ 
11:  end if
12:   $\mathcal{L} \leftarrow \lambda_E \text{MAE}_E + \lambda_F \text{MAE}_F + \lambda D_{\text{KL}}(\alpha, \beta)$  {Loss includes MAEs and KL
    divergence term from (77)}
13: end while

```

INFERENCE WITH BLIP. Uncertainty quantification proceeds via multiple stochastic forward passes using Monte Carlo (MC) sampling. Crucially, the inference network is evaluated only once to extract the variational parameters

Algorithm 6 BLIP Inference with Uncertainty Quantification

```

1:  $\alpha, \beta \leftarrow E_\psi(h^0, a_{ij})$  {Encode inputs to obtain VAD coefficients once}
2: energies  $\leftarrow []$ , forces  $\leftarrow []$ 
3: for each MC sample  $s = 1, \dots, S$  do
4:    $h^0 \leftarrow$  initial node features
5:   for layer  $l = 0, \dots, L - 1$  do
6:      $h^{l+1} \leftarrow \text{MessagePassing}(h^l, a_{ij}, \alpha^l, \beta^l)$  {Sample weights via local
       reparametrisation (Alg. 4)}
7:   end for
8:   Compute energy  $E^{(s)}$  from  $h^{1:L}$  by pooling
9:   if field is conservative then
10:     $F^{(s)} \leftarrow -\nabla E^{(s)}$ 
11:   else
12:    Compute  $F^{(s)}$  directly from  $h^L$ 
13:   end if
14:   Append  $E^{(s)}$  to energies,  $F^{(s)}$  to forces
15: end for
16: Compute mean and variance statistics over energies and forces

```

α, β , which are then reused across all MC samples, substantially reducing computational overhead. When only predictive means are required, a single deterministic forward pass using the posterior mean weights (i.e., without stochastic sampling) suffices, matching the speed of a standard MLIP. More accurate moment-matching approximations could be used for the output distributions, but this is left for future work.

4.10 COMPUTATIONAL COST ANALYSIS

A central motivation of BLIP is to achieve high-quality uncertainty estimates and accurate mean predictions with minimal additional computational overhead. This section quantifies the training, inference, sampling, and memory costs of BLIP relative to deep ensembles. Table 15 summarises key statistics including parameter counts and FLOPs for training and inference.

Algorithm 7 BLIP Inference with Mean Energy and Forces Prediction Only

```

1: for layer  $l = 0, \dots, L - 1$  do
2:    $h^{l+1} \leftarrow \text{MessagePassing}(h^l, a_{ij}, \mathbf{0}, \mathbf{0})$  {Use mean weights, no perturbation}
3: end for
4: Compute energy  $E$  from  $h^{1:L}$  by pooling
5: if field is conservative then
6:    $\mathbf{F} \leftarrow -\nabla E$ 
7: else
8:   Compute  $\mathbf{F}$  directly from  $h^L$ 
9: end if

```

Table 15: Training and inference cost. Parameter counts and FLOPs for the ORBv3-inf-omat24 (Base) model, BLIP, and ensembles.

		Base	Base + BLIP	$N \times$ Ensemble Base
	Params	21.02	21.05	$21.02 \times N$
	Params increase from Base	$\times 1$	$\times 1.001$	$\times N$
<i>Training</i>	FLOPs / atom ($\times 10^{12}$)	1.65	3.1	$1.65 \times N$
	FLOPs increase from Base	$\times 1$	$\times 1.9$	$\times N$
<i>Inference</i>	FLOPs / atom ($\times 10^{12}$)	1.65	1.65	$1.65 \times N$
	FLOPs increase from Base	$\times 1$	$\times 1$	$\times N$

TRAINING COST. BLIP trains a single backbone jointly with a lightweight inference network E whose parameter count is typically 0.1% to 5% of the main model. Denoting by C_{main} the FLOPs for training the backbone and by C_{inf} those for the inference network, the total training cost is:

$$C_{\text{BLIP}} = C_{\text{main}} + C_{\text{inf}} \approx (2 + \epsilon) C_{\text{main}},$$

where $\epsilon = C_{\text{inf}}/C_{\text{main}} \ll 1$. The factor of two arises from the local reparametrisation trick, which requires computing both the mean and variance activations. By contrast, an ensemble of N independently trained models incurs:

$$C_{\text{ens}} = N C_{\text{main}},$$

which is N times the cost of a single model. For typical ensemble sizes ($N = 5$), this translates to roughly a $5\times$ increase in both computation and memory footprint.

INFERENCE COST. At inference time, BLIP performs a single forward pass through the backbone with the variational dropout coefficients set to zero, incurring **no additional overhead** relative to a standard deterministic model. An ensemble of N models, by contrast, requires N separate forward passes, with cost scaling linearly in N .

SAMPLING COST. Each posterior sample requires the same forward FLOPs as during training. Techniques such as vectorial mapping, moment matching, or moment linearisation could substantially reduce the cost of estimating predictive distribution moments, and VAD accomodates in easily (Kruiff et al. 2026a).

BAYESIAN LOW-RANK ADAPTATION OF LARGE SCALE MODELS

Low-Rank Adaptation (LoRA) has become the standard for fine-tuning large pre-trained models at reduced computational cost. However, its low-rank point-estimate updates limit expressiveness, leave a persistent gap relative to full fine-tuning accuracy, and provide no built-in uncertainty quantification, limiting its applicability in settings where reliability matters as much as accuracy. In this chapter, we introduce Bayesian Low-Rank Adaptation (BaLoRA), a Bayesian extension of LoRA with a novel input-adaptive Bayesian parameterization of LoRA matrices that adds minimal parameters and compute. Surprisingly, not only does the Bayesian extension yield well-calibrated uncertainty estimates, but the adaptive noise injection underlying our approach also significantly improves prediction accuracy, narrowing the gap with full fine-tuning across both natural language reasoning and vision tasks. When applied to band gap prediction in metal-organic frameworks, BaLoRA produces zero-shot test-time uncertainty estimates that correlate more strongly with model error than a trained ensemble of LoRA models, and improve monotonically with compute without sacrificing accuracy.

5.1 INTRODUCTION

Large pre-trained models have demonstrated remarkable generalisation capabilities across a broad spectrum of tasks, ranging from natural language processing (C. Qin et al. 2023; Touvron, Lavril, et al. 2023) and multi-modal applications (H. Liu et al. 2023) to atomistic simulations (Wood et al. 2025; Shoghi et al. 2024; Kang et al. 2023). Adapting such models to specific down-

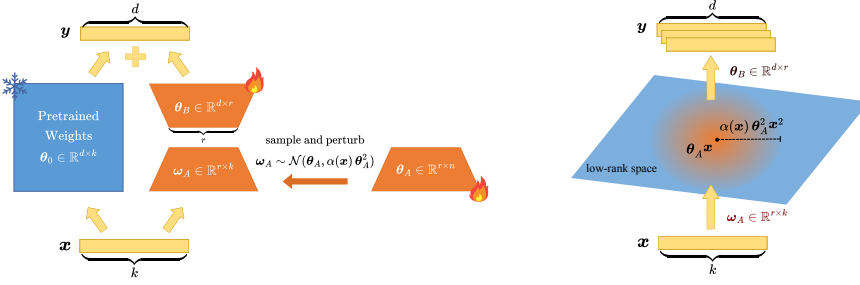


Figure 21: Overview of BaLoRA. *Left:* The reduction matrix is treated as a random variable with input-adaptive Gaussian noise while the pretrained weights remain frozen. *Right:* The resulting uncertainty ellipsoid surrounds the LoRA estimate within the low-rank subspace, with its size determined by the input activation.

stream tasks commonly relies on full fine-tuning (FT), a procedure that updates all model parameters using task-specific data. Nonetheless, as model sizes continue to grow, FT becomes increasingly impractical owing to its substantial computational and memory demands. Even in cases where FT is feasible, deploying several independently fine-tuned instances poses significant challenges, since each one retains the complete model footprint.

Parameter-efficient fine-tuning (PEFT) (Houlsby et al. 2019) addresses these limitations by updating only a small subset of the model parameters while keeping the remainder frozen, thus improving computational efficiency without substantially degrading performance. Low-Rank Adaptation (LoRA) (E. J. Hu et al. 2022) represents one of the most widely adopted PEFT methods; it constrains parameter updates to a low-rank structure, capturing task-relevant changes with considerably fewer trainable parameters. Nevertheless, LoRA and related approaches share an intrinsic limitation: their low-rank, point-estimate updates reduce representational expressiveness and leave a noticeable gap relative to full fine-tuning (S.-Y. Liu et al. 2024). Furthermore, these methods offer no mechanism for uncertainty quantification, which is indispensable in safety-critical and data-scarce settings. This shortcoming is particularly consequential in scientific domains such as materials science and weather forecasting, where reliable uncertainty estimates are a fundamental require-

ment (Wood et al. 2025; Bodnar et al. 2025; Coscia, Haan, et al. 2025; Coscia, Welling, et al. 2025).

Motivated by the limitations of existing PEFT methods and the increasing demand for principled uncertainty estimates in scientific applications, we introduce BaLoRA, a Bayesian extension of LoRA that simultaneously addresses both challenges. BaLoRA proposes a novel input-adaptive Bayesian reparametrisation of the LoRA reduction matrix, whereby its entries are treated as random variables rather than fixed point estimates. This formulation enables uncertainty to be naturally encoded during training through adaptive noise injection, which steers the reduction matrix towards inputs associated with higher uncertainty, thereby improving task performance. At inference time, BaLoRA can operate in deterministic mode by merging the adapter weights as in standard LoRA, with no additional latency, or switch to Bayesian mode whenever uncertainty estimates are needed. The main contributions of this chapter are summarised as follows: first, we propose a stochastic PEFT method built on an input-adaptive Bayesian reparametrisation of the LoRA reduction matrix, in which input-dependent noise injection serves as an implicit regulariser that facilitates training and yields performance improvements, while simultaneously enabling principled uncertainty quantification. Second, we derive a low-rank local reparametrisation trick that exploits the LoRA factorisation to draw exact samples from the posterior predictive in the low-dimensional latent space, avoiding the need to materialise the full output covariance and preserving the same computational scaling as standard LoRA. Finally, we attain state-of-the-art performance on commonsense reasoning across 6 out of 8 benchmarks (Llama-3-8B) and on image classification (ViT-L/16), and demonstrate the practical utility of BaLoRA’s uncertainty estimates in a real-world scientific setting (MOF property prediction), where BaLoRA surpasses a trained LoRA ensemble with a single fine-tuning run.

5.2 BACKGROUND AND RELATED WORKS

This section introduces the relevant background on Parameter-Efficient Fine-Tuning, Low-Rank Adapters, Bayesian modelling, and Uncertainty Quantification, all of which underpin the formulation of the proposed method presented later in this chapter.

5.2.1 *Parameter-Efficient Fine-Tuning (PEFT)*

Parameter-efficient fine-tuning (PEFT) (Houlsby et al. 2019) enables the adaptation of large pre-trained models to downstream tasks at low computational cost by updating only a small fraction of the total parameters. Two principal categories can be identified. *Prompt-based* methods (Lester et al. 2021; Razdaibiedina et al. 2023; X. Liu, Y. Zheng, et al. 2024) prepend learnable soft tokens to the model input, optimising only these additional vectors while all original weights remain frozen. *Adapter-based* methods (Houlsby et al. 2019; Mahabadi et al. 2021) instead insert small trainable modules into the frozen architecture, offering strong downstream performance and stable training dynamics. However, the majority of adapter-based approaches introduce inference overhead, which limits their practical deployability.

Low-Rank Adapters such as LoRA (E. J. Hu et al. 2022) add no extra inference latency, grounded in the observation that fine-tuning updates possess low intrinsic rank (Aghajanyan et al. 2021). Given that LoRA can underperform full fine-tuning across diverse tasks, several extensions have been put forward. DoRA (S.-Y. Liu et al. 2024) decomposes weight updates into magnitude and directional components, applying a LoRA-style update exclusively to the directional part. MoRA (T. Jiang et al. 2024) projects inputs into a compressed space, performs transformations via a higher-rank matrix, and reconstructs the result back to the original dimensionality. HiRA (Q. Huang et al. 2025) employs a Hadamard product to retain high-rank update information, thereby augmenting the expressive capacity of the adapted model.

5.3 METHODS

In this chapter, we introduce BaLoRA, a novel parameter-efficient fine-tuning strategy that builds upon VAD (Coscia, Haan, et al. 2025) and extends the LoRA method by naturally encoding uncertainty during both training and inference. The key contribution of BaLoRA resides in the Bayesian reparametrization of the learning process, wherein an adaptive variance is incorporated into the LoRA reduction matrix (see Figure 21). During training, the input-adaptive noise functions as a stochastic regulariser: larger activations in ω_A induce proportionally larger perturbations, guiding the model towards a more robust

low-rank representation. Crucially, this adaptivity ensures that perturbations are concentrated where activations are largest, preventing overfitting precisely in the most active directions while leaving stable directions unaffected. At test time, BaLoRA can be deployed either in deterministic inference mode, with no additional latency over LoRA, or in Bayesian mode when uncertainty quantification is required.

5.3.1 Low-Rank Adaptation

LoRA (E. J. Hu et al. 2022) is a fine-tuning strategy that freezes the pre-trained model weights and injects trainable rank decomposition matrices into each linear layer of a neural network. For a pre-trained weight matrix $\theta_0 \in \mathbb{R}^{k \times d}$, the update is constrained to a low-rank factorization $\theta_B \theta_A$, where $\theta_A \in \mathbb{R}^{r \times d}$ is the *reduction* matrix, and $\theta_B \in \mathbb{R}^{k \times r}$ is the *reconstruction* matrix, with rank $r \ll \min(d, k)$. Given input $x \in \mathbb{R}^d$, the forward pass is given by:

$$y = \theta_0 x + \theta_B \theta_A x. \quad (99)$$

Typically, θ_A is initialized from an isotropic normal distribution with a small standard deviation, while θ_B is initialized to zero, ensuring the adapter contributes no signal at the start of training. Only θ_A and θ_B are optimized; at inference time, the adapter is merged into the pre-trained weights as $\theta = \theta_0 + \theta_B \theta_A$, incurring no additional inference cost.

5.3.2 Bayesian Adaptive LoRA

To incorporate uncertainty into LoRA, we treat the entries of the reduction matrix as random variables rather than fixed parameters. We denote by ω_A the random reduction matrix. To construct it, we perturb each entry of the deterministic reduction matrix θ_A with input-dependent Gaussian noise, following the approach of VAD (Coscia, Haan, et al. 2025):

$$\begin{aligned} \omega_{A;ij} &= \theta_{A;ij} + \sqrt{\alpha(x)\theta_{A;ij}^2} \epsilon_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(0, 1). \\ \iff q(\omega_A \mid x) &= \prod_{i=1}^r \prod_{j=1}^d \mathcal{N}(\theta_{A;ij}, \alpha(x)\theta_{A;ij}^2) \end{aligned} \quad (100)$$

Here, $\alpha(x)$ is a lightweight inference network that modulates the overall scale of the variance as a function of the input x . Intuitively, weights with larger magnitude carry proportionally more uncertainty, with α governing how strongly the input drives that uncertainty.

Under this posterior, since θ_B is deterministic, the predictive distribution of the output given the input is exactly Gaussian (see Appendix 5.6.1 for derivations):

$$y \sim \mathcal{N}\left(\theta_0 x + \theta_B \theta_A x, \alpha \theta_B \text{diag}(\theta_A^2 x^2) \theta_B^\top\right), \quad (101)$$

where squares are applied element-wise. Importantly, the mean coincides with the standard LoRA forward pass, meaning that at inference time the weights can still be merged as $\theta = \theta_0 + \theta_B \theta_A$, incurring no additional inference cost. Furthermore, owing to the multiplicative noise formulation, the only additional parameters introduced with respect to LoRA originate from the inference network, which is lightweight and practically negligible in size.

TRAINING AND KL DIVERGENCE. Variational Adaptive Dropout trains the Bayesian model by maximising the Evidence Lower Bound, which balances data fit (the expected log-likelihood term) against model complexity (the KL divergence between the posterior and the prior). The ELBO takes the form:

$$\mathcal{L} = \mathbb{E}_{\omega_A \sim q(\omega_A | x)} [\log p(y | \omega_A, x)] - D_{KL} [q(\omega_A | x) | p(\omega_A)]. \quad (102)$$

In practice, a single Monte Carlo sample from the variational posterior at each training iteration provides a reliable approximation of eq. (102). We employ the same weakly informed prior as Coscia, Haan, et al. (2025):

$$p(\omega_A) = \prod_{i=1}^r \prod_{j=1}^d \mathcal{N}\left(0, \frac{p}{1-p} \theta_{A:ij}^2\right), \quad (103)$$

which admits the interpretation of applying dropout to the network with dropout probability p . Under this choice, the KL divergence between posterior and prior has a closed form:

$$D_{KL}[q(\omega_A | x) | p(\omega_A)] = \frac{1}{2} \left(\frac{(\alpha + 1)(1 + p)}{p} + \log \frac{p}{1-p} - \log(\alpha) - 1 \right), \quad (104)$$

which depends only on the scalar α and the dropout rate p , making it inexpensive to evaluate and to differentiate through. A complete derivation is provided in Appendix 5.6.2.

5.3.3 Low-Rank Local Reparametrization Trick

The local reparametrization trick (Durk P Kingma et al. 2015) was introduced to enable efficient sampling from the predictive distribution of Bayesian linear layers. For a single linear layer with input x and weight matrix $\omega \sim \mathcal{N}(\theta, \alpha\theta^2)$, the output y follows a normal distribution:

$$y \sim \mathcal{N}(\gamma, \text{diag}(\delta)), \quad (105)$$

where each output element y_j can be expressed as

$$y_j = \gamma_j + \sqrt{\delta_j} \epsilon_j, \quad \gamma_j = \sum_i x_i \theta_{ij}, \quad \delta_j = \alpha \sum_i x_i^2 \theta_{ij}^2, \quad \epsilon_j \sim \mathcal{N}(0, 1). \quad (106)$$

Accordingly, rather than drawing samples from the weight matrix ω directly, one samples the output y from a Gaussian whose mean and variance are computed analytically, reducing both variance and computational cost to two matrix multiplications.

LOW-RANK SAMPLING. In BaLoRA, a straightforward application of the local reparametrization trick is not directly viable. Since the stochastic reduction matrix θ_A is followed by the deterministic projection θ_B , the posterior predictive covariance of the full LoRA update is $\Sigma = \alpha \theta_B \text{diag}(\theta_A^2 x^2) \theta_B^\top$, which has size $\mathcal{O}(Bk^2)$ for a batch of size B and output dimension k . Explicitly constructing and sampling from this matrix is computationally prohibitive, even for moderate values of k .

We instead exploit the low-rank structure of the covariance to derive an efficient sampling procedure. Define the element-wise variance vector $\mathbf{d}(x) = \alpha \theta_A^2 x^2 \in \mathbb{R}^r$. The covariance then admits the factorization:

$$\Sigma = \theta_B \text{diag}(\mathbf{d}(x)) \theta_B^\top = \mathbf{A}(x) \mathbf{A}(x)^\top, \quad \mathbf{A}(x) = \theta_B \text{diag}(\sqrt{\mathbf{d}(x)}), \quad (107)$$

where we used the identity $\text{diag}(\mathbf{d}) = \text{diag}(\sqrt{\mathbf{d}}) \text{diag}(\sqrt{\mathbf{d}})$. Drawing $\epsilon_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r)$, an exact sample from the posterior predictive is then obtained by (see Appendix 5.6.1 for derivations):

$$\mathbf{y} = \theta_0 \mathbf{x} + \theta_B \theta_A \mathbf{x} + \theta_B (\sqrt{\mathbf{d}(\mathbf{x})} \odot \epsilon_d), \quad (108)$$

where $\odot$ denotes element-wise multiplication. Crucially, the noise is sampled in the low-dimensional space $\mathbb{R}^r$ before being projected up by θ_B , so the full covariance matrix is never materialised. The resulting complexity is $\mathcal{O}(Bkr)$ in time and $\mathcal{O}(Br + Bk)$ in memory, matching the scaling of standard LoRA inference and adding only a constant factor overhead. This contrasts with the $\mathcal{O}(Bk^2)$ cost of naively sampling the full covariance, making BaLoRA practical even at large output dimensions where $k \gg r$.

5.3.4 Geometric Interpretation and Implicit Regularisation of BaLoRA

It is instructive to examine BaLoRA from a geometric perspective. Standard LoRA constrains the weight update to a low-dimensional subspace of dimension $r \ll k$: the update $\theta_B \theta_A \mathbf{x}$ first projects the input onto the r -dimensional row space of θ_A , and then lifts the result back to the k -dimensional output space via θ_B . The expressivity of the adaptation is therefore deliberately restricted to the column space of θ_B , denoted $\mathcal{V} = \text{col}(\theta_B) \subset \mathbb{R}^k$.

BaLoRA preserves this structure while equipping it with an uncertainty estimate. The noise term $\theta_B (\sqrt{\mathbf{d}(\mathbf{x})} \odot \epsilon_d)$ defines a random perturbation that, by construction, also lies in $\mathcal{V}$. Geometrically, the predictive distribution can be understood as placing an *input-adaptive ellipsoid* around the LoRA point estimate, confined entirely to the subspace $\mathcal{V}$:

$$\mathbf{y} = \underbrace{\theta_0 \mathbf{x}}_{\text{pretrained}} + \underbrace{\theta_B \theta_A \mathbf{x}}_{\text{LoRA update}} + \underbrace{\theta_B (\sqrt{\mathbf{d}(\mathbf{x})} \odot \epsilon_d)}_{\text{uncertainty in } \mathcal{V}}. \quad (109)$$

The shape of this ellipsoid (Figure 21 Left) is governed by the diagonal matrix $\text{diag}(\mathbf{d}(\mathbf{x})) = \alpha(\mathbf{x}) \text{diag}(\theta_A^2 \mathbf{x}^2)$, which carries two geometrically meaningful properties. First, the ellipsoid is *aligned with the LoRA basis*: its principal axes correspond to the columns of θ_B , so uncertainty is expressed along the same

directions as the adaptation itself. Second, the ellipsoid is *input-adaptive*: its extent along each axis scales with $\mathbf{d}(x)$, meaning that inputs producing large activations in the reduction matrix θ_A are assigned wider uncertainty, while inputs lying in the null space of θ_A carry no uncertainty.

We postulate that this geometric structure facilitates learning in two complementary ways. First, by confining stochastic perturbations to $\mathcal{V}$, noise is prevented from propagating into the pretrained representations outside the adapted subspace, thereby preserving the stability of directions the model has not chosen to adapt. Second, within $\mathcal{V}$, the input-adaptive scaling takes a multiplicative noise form, inducing larger perturbations precisely where activations are strongest; this discourages overconfident updates in the most active directions and consequently acts as an implicit regulariser. Together, these effects allow $\alpha(x)$ to focus on identifying genuinely ambiguous inputs within the adapted subspace.

5.4 EXPERIMENTS

To evaluate BaLoRA, we design experiments spanning three domains: natural language, computer vision, and materials science. We first demonstrate that introducing a Bayesian framework enhances LoRA’s optimization behavior under fixed hyperparameter settings, assessed on commonsense reasoning benchmarks. On these tasks, we additionally compare BaLoRA against a range of Parameter-Efficient Fine-Tuning baselines using LLaMA2-7B and LLaMA3-8B, achieving leading performance across multiple settings, and we report an ablation study examining the role of the prior probability. We subsequently broaden our evaluation to computer vision, benchmarking BaLoRA against alternative PEFT methods when adapting a Vision Transformer to several image classification datasets. We further examine BaLoRA’s uncertainty calibration and carry out a runtime and computational overhead comparison with other PEFT approaches. Finally, we turn to materials science, applying BaLoRA to MOF property prediction, where it not only surpasses standard LoRA in predictive accuracy but also yields superior uncertainty estimates compared to a LoRA ensemble. Experimental details and hyperparameters are provided in Appendix 5.7.

Table 16: Accuracy gains from Bayesian adaptation. Accuracy for standard LoRA and BaLoRA under equivalent dropout coefficients. Blue and red indicate percentage improvements and decreases relative to LoRA, respectively.

Model	Method	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG
Llama-2-7B	LoRA	69.80	79.90	79.50	64.70	79.80	81.00	83.60	82.60
	BaLoRA	71.59 ^{+2.6%}	83.73 ^{+4.8%}	77.12 ^{-3.0%}	69.45 ^{+7.3%}	84.39 ^{+5.8%}	81.40 ^{+0.5%}	87.73 ^{+4.9%}	82.72 ^{+0.1%}
Llama-3-8B	LoRA	70.80	85.20	79.90	71.20	84.20	79.00	91.70	84.30
	BaLoRA	75.29 ^{+6.3%}	88.52 ^{+3.9%}	80.45 ^{+0.7%}	78.58 ^{+10.4%}	89.73 ^{+6.6%}	84.20 ^{+6.6%}	95.46 ^{+4.1%}	85.24 ^{+1.1%}

5.4.1 Commonsense Reasoning

We benchmark against prompt-based approaches, specifically Prompt Tuning (Lester et al. 2021) and P-tuning (X. Liu, Ji, et al. 2022), as well as low-rank adaptation methods, namely LoRA (E. J. Hu et al. 2022), DoRA (S.-Y. Liu et al. 2024), MORA (T. Jiang et al. 2024), and HiRA (Q. Huang et al. 2025). All experiments are conducted on two open-source Large Language Models (LLMs): Llama-2-7B (Touvron, Lavril, et al. 2023) and Llama-3-8B (Grattafiori et al. 2024). The commonsense reasoning evaluation covers 8 sub-tasks, each with its own predefined train/test split. Unless noted otherwise, we adopt the experimental setup and hyperparameter configuration of Q. Huang et al. (2025). We evaluate commonsense reasoning following the evaluation protocol of Z. Hu et al. (2023) and S.-Y. Liu et al. (2024), where accuracy is used as the performance metric. For each query, the model’s response is determined by identifying the first occurrence of a task-specific keyword (e.g., “true” or “false” for BoolQ); if no such keyword is found, the response is marked incorrect.

BALoRA TRAINING DYNAMICS IMPROVE LoRA PERFORMANCE. To isolate the effect of the Bayesian components, we conduct a controlled comparison between LoRA and BaLoRA under identical settings. We use the same weight initialization, dropout probability, and hyperparameters, differing only in the stochastic adaptive injection and KL regularization introduced by BaLoRA. As shown in Table 16, BaLoRA consistently outperforms LoRA by an average margin of $\sim 2.9\%$ on Llama-2 and $\sim 5\%$ on Llama-3, supporting the hypothesis that exploiting geometric structure facilitates optimization.

Table 17: Commonsense reasoning accuracy. Accuracy across PEFT methods and eight benchmarks. ChatGPT, LoRA, and DoRA results are from (Q. Huang et al. 2025); bold and underlined values indicate the best and second-best results, respectively.

Model	Method	Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG
ChatGPT	-	-	73.10	85.40	68.50	79.90	89.80	74.80	78.50	66.10
Llama-2-7B	Prompt Tuning	0.0012	55.93	12.35	30.50	6.06	8.63	9.40	6.91	40.57
	P-Tuning	0.7428	58.75	36.02	0.20	0.17	1.98	0.80	0.01	0.00
	LoRA	0.8256	69.80	79.90	79.50	64.70	79.80	81.00	83.60	82.60
	DoRA	0.8256	71.80	<u>83.70</u>	76.00	68.20	83.70	82.40	<u>89.10</u>	82.60
	MoRA	0.8241	<u>72.17</u>	80.79	<u>79.53</u>	71.42	85.31	81.20	29.09	80.19
	HiRA	0.8256	71.22	83.35	<u>79.53</u>	73.81	86.74	84.60	88.12	83.98
	BaLoRA	0.8427	72.69	84.93	80.45	<u>72.10</u>	<u>85.56</u>	85.40	91.23	<u>83.66</u>
Llama-3-8B	Prompt Tuning	0.0010	56.85	45.05	36.13	31.57	32.74	29.20	14.01	50.12
	P-Tuning	0.6240	59.97	11.64	8.19	7.42	8.63	9.60	1.77	37.65
	LoRA	0.7002	70.80	85.20	79.90	71.20	84.20	79.00	91.70	84.30
	DoRA	0.7002	74.60	89.30	79.90	80.40	90.50	85.80	<u>95.50</u>	85.60
	MoRA	0.6997	74.28	87.43	80.71	79.61	91.16	85.60	43.53	86.74
	HiRA	0.7002	<u>75.40</u>	<u>89.70</u>	<u>81.15</u>	82.90	93.27	<u>88.32</u>	95.36	<u>87.70</u>
	BaLoRA	0.7145	76.42	89.99	81.78	<u>80.90</u>	<u>91.20</u>	88.40	96.11	88.40

BaLoRA ACHIEVES STATE-OF-THE-ART ON COMMONSENSE REASONING. Table 17 summarizes accuracy across eight commonsense reasoning benchmarks. BaLoRA ranks first on five out of eight tasks for Llama-2-7B and six out of eight tasks for Llama-3-8B, while remaining within a comparable parameter budget to all low-rank baselines. The additional trainable parameter count (*Params* in Table 17) is attributed to the inference network predicting α , and is negligible relative to the baseline models. Against the closest competitor, HiRA, BaLoRA delivers steady gains across the majority of sub-tasks, suggesting that the Bayesian geometric prior confers an advantage over purely deterministic rank decomposition. Taken together, these results establish BaLoRA as a robust and parameter-efficient alternative to existing PEFT methods. To the best of our knowledge, this is the first method where improvements are driven by principled uncertainty modeling rather than by increased model capacity.

BaLoRA IS ROBUST ACROSS PRIOR PROBABILITIES. In Figure 22 we examine the sensitivity of BaLoRA to the choice of prior probability p in the KL divergence loss of eq. (104) for Llama3 across BoolQ, PIQA, SIQA and WinoG datasets (see Appendix 5.7 for Llama2 and additional datasets). Across

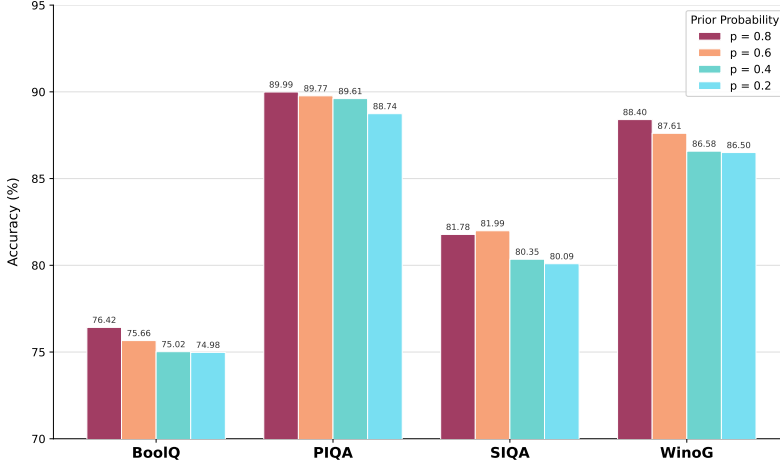


Figure 22: Prior-probability ablation. Accuracy on LLaMA-3 across benchmark tasks for different values of the prior probability p in the KL-divergence loss. Results are robust across the tested values of p .

both Llama-2 and Llama-3, performance remains largely stable over the full range of tested values, with larger values of p showing marginal accuracy gains across multiple sub-tasks. The variance in average accuracy across p values is narrow, indicating that BaLoRA is robust to this hyperparameter.

5.4.2 Visual Perception

Having established BaLoRA’s advantage on language tasks, we turn to visual perception to assess the generality of our approach across domains. We benchmark BaLoRA against LoRA and DoRA, the two most established low-rank adaptation methods in the vision literature, by fine-tuning a ViT-L/16 pretrained on ImageNet-21K (Dosovitskiy 2020) across four image classification benchmarks: CIFAR-10, CIFAR-100, Oxford-IIIT Pets, and Oxford Flowers-102. We follow the standard experimental protocol of Dosovitskiy (2020) with minor learning rate adjustments, applying adapters exclusively to the query and value projection matrices as is conventional in the timm library. We set $r=8$, $\alpha=16$ for CIFAR datasets and $r=16$, $\alpha=32$ otherwise; full hyperparameter

Table 18: Image-classification accuracy. Accuracy across PEFT methods for ViT-L/16 on four datasets. Full fine-tuning is included as a reference; bold and underlined values indicate the best and second-best results, respectively.

Method	CIFAR10	CIFAR100	Oxford-IIIT Pets	Oxford Flowers-102
Full Fine-tuning	99.15	93.25	94.67	99.61
LoRA	98.93	<u>93.59</u>	93.96	99.44
DoRA	<u>99.03</u>	93.44	<u>94.01</u>	<u>99.48</u>
BaLoRA	99.04	93.62	94.07	99.61

Table 19: Image-classification calibration. Expected calibration error across PEFT methods for ViT-L/16 on four datasets. Bold and underlined values indicate the best and second-best results, respectively; values are percentages.

Method	CIFAR10	CIFAR100	Oxford-IIIT Pets	Oxford Flowers-102
LoRA	0.81	<u>4.25</u>	3.39	0.62
DoRA	<u>0.73</u>	4.45	3.28	0.70
BaLoRA	0.71	4.12	<u>3.33</u>	<u>0.63</u>

details are provided in Appendix 5.8. Beyond classification accuracy, we additionally report Expected Calibration Error (ECE), a principled measure of predictive reliability, to assess whether BaLoRA’s stochastic training objective yields more calibrated models than its deterministic counterparts.

BALoRA IS ACCURATE AND CALIBRATED ON IMAGE CLASSIFICATION. Classification accuracy across four benchmarks is reported in Table 18. BaLoRA consistently ranks first among all PEFT methods, matching or approaching full fine-tuning performance. Most notably, BaLoRA achieves state-of-the-art results on CIFAR100, surpassing even the fully fine-tuned model, and it closes the performance gap entirely on Oxford Flowers-102, matching the full fine-tuning ceiling of 99.61% accuracy. Against the strongest PEFT baseline, DoRA, BaLoRA delivers consistent improvements across all four datasets. These results demonstrate that BaLoRA’s Bayesian inductive bias is not specific to

the language domain but transfers naturally to visual perception, providing a robust and parameter-efficient alternative to deterministic low-rank adaptation.

Beyond predictive accuracy, well-calibrated confidence estimates are essential for trustworthy deployment in real-world settings. Table 19 reports the Expected Calibration Error across the four benchmarks. BaLoRA achieves the lowest ECE on two out of four datasets and ranks second on the remaining two, consistently outperforming or matching both LoRA and DoRA without any post-hoc calibration procedure. This indicates that the stochastic adaptive injection and KL regularization jointly act as an implicit regularizer, discouraging overconfident predictions.

BALoRA ADDS MINIMAL COMPUTATIONAL COST OVER LoRA. Table 20 compares memory and compute overhead across methods during training¹. BaLoRA introduces a modest increase in trainable parameters (1.14M vs. 0.80M for LoRA) and a $\times 1.29$ FLOPs overhead relative to full fine-tuning, a direct consequence of the stochastic sampling during the forward pass. Peak memory consumption remains well below that of full fine-tuning and DoRA, confirming that BaLoRA operates comfortably within the parameter-efficient regime despite its Bayesian formulation.

Table 20: Training resource comparison. Trainable parameters, peak memory, and FLOPs for PEFT methods on CIFAR-100 with ViT-L/16. Measurements use a single H100 94 GB GPU and one image input.

Method	Params (M)	Peak Memory Fwd (Gb)	Peak Memory Bwd (Gb)	FLOPs (G)
Full Fine-tuning	305	11.04	11.16	$\times 1$
LoRA	0.80	8.13	8.15	$\times 1.003$
DoRA	0.85	10.55	10.57	$\times 1.003$
BaLoRA	1.14	9.73	9.76	$\times 1.29$

¹ Inference statistics are omitted as all models, BaLoRA included, can merge adapter weights with the pre-trained weights, resulting in zero additional latency or memory cost.

5.4.3 *Materials Property Prediction*

Having evaluated BaLoRA in language and vision domains, we extend to a scientific setting to assess both predictive accuracy and uncertainty quantification, a capability that deterministic low-rank methods lack in regression-based settings. We fine-tune MOFTransformer (Kang et al. 2023) on the QMOF dataset (Rosen, Iyer, et al. 2021) for bandgap prediction, an important task in designing MOFs for energy storage applications (Rosen, Fung, et al. 2022). Based on preliminary results, we apply adapters only to MLP layers with $r=64$ and $\alpha=128$. We compare BaLoRA against an ensemble of LoRA models, a strong uncertainty quantification baseline requiring multiple adapters, while BaLoRA provides calibrated uncertainty from a single adapter. Importantly, training multiple adapters requires multiple independent fine-tuning runs, whereas BaLoRA requires only one. We evaluate predictive accuracy using Mean Absolute Error (MAE) and uncertainty quality via Spearman correlation between predicted uncertainty and actual error. Full hyperparameter details are provided in Appendix 5.7.

BALORA IS ACCURATE AND CALIBRATED ON BANDGAP PREDICTION. Figure 25 reports MAE and Spearman correlation on the QMOF benchmark. BaLoRA consistently outperforms DoRA and LoRA across both metrics. In terms of accuracy, BaLoRA achieves an MAE of 0.266 eV, narrowing the gap to full fine-tuning (0.224 eV) while remaining significantly more parameter-efficient (using only 6.75% of trainable parameters). Notably, its predictions match ground truth values more closely in the low bandgap regime (< 2 eV), which is particularly important since lower bandgaps correspond to higher conductivity and are critical for applications such as photocatalysis (Cao et al. 2023). For uncertainty quantification, BaLoRA achieves the highest Spearman correlation (0.346) between predicted uncertainty and actual error, outperforming ensemble DoRA (0.259) and LoRA (0.309). Interestingly, performing multiple forward passes at test time further improves uncertainty estimates without requiring additional training (Figure 25). Nevertheless, all methods exhibit only moderate correlations, underscoring the intrinsic difficulty of reliable uncertainty estimation in this task. This indicates that while BaLoRA

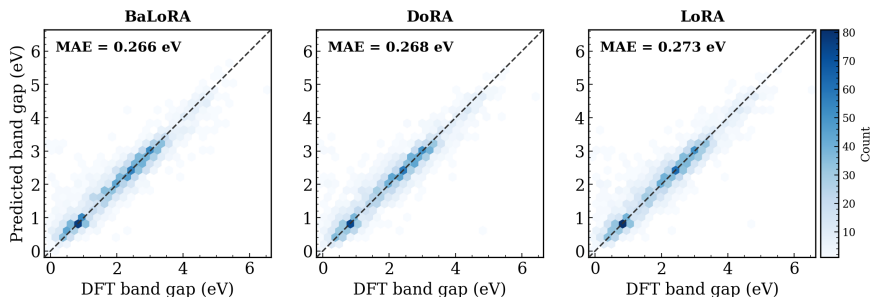


Figure 23: Bandgap prediction on QMOF. Performance in eV for PEFT methods using MOFTransformer.

improves calibration, bandgap prediction for MOFs remains a challenging setting for uncertainty-aware modeling.

5.5 FINAL DISCUSSION

We presented BaLoRA, a Bayesian extension of Low-Rank Adaptation that addresses two fundamental limitations of existing PEFT methods: the expressiveness gap relative to full fine-tuning, and the lack of principled uncertainty quantification. By introducing an input-adaptive Bayesian reparametrization of the LoRA reduction matrix, BaLoRA encodes uncertainty during training via adaptive noise injection, which simultaneously acts as an implicit regularizer and enables calibrated predictions at test time. A low-rank local reparametrization trick ensures that the posterior distribution is sampled exactly in the low-dimensional latent space, matching the computational scaling of standard LoRA and incurring no additional inference overhead in deterministic mode.

Across commonsense reasoning, visual perception, and materials property prediction, BaLoRA consistently outperforms LoRA and its deterministic variants while remaining within a comparable parameter and FLOPs budget. On commonsense reasoning, BaLoRA achieves state-of-the-art accuracy on the majority of sub-tasks for both Llama-2-7B and Llama-3-8B, driven by principled uncertainty training rather than increased model capacity. On image classification, it matches full fine-tuning accuracy on Oxford Flowers-102 and achieves strong calibration across all benchmarks without post-hoc

correction. Finally, on bandgap prediction for metal-organic frameworks, BaLoRA produces zero-shot uncertainty estimates that correlate more strongly with model error than a trained ensemble of LoRA models, while requiring only a single fine-tuning run. These results suggest that Bayesian inductive biases, when carefully integrated into the geometry of low-rank adapters, offer complementary and additive advantages over purely deterministic approaches

CHAPTER APPENDIX

5.6 PROOFS AND DERIVATIONS

This appendix is devoted to the formal proofs introduced in the main text.

5.6.1 Predictive Distribution and Uncertainty Estimates

We begin by deriving the posterior predictive distribution of the output induced by the stochastic LoRA update (Figure 21).

Proposition 5.1 (Predictive Distribution). *Under the posterior*

$$q(\boldsymbol{\omega}_A \mid \mathbf{x}) = \prod_{i=1}^r \prod_{j=1}^d \mathcal{N}(\theta_{A;ij}, \alpha(\mathbf{x}) \theta_{A;ij}^2), \quad (110)$$

and with deterministic $\boldsymbol{\theta}_B$, the output

$$\mathbf{y} = \boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\omega}_A \mathbf{x} \quad (111)$$

is Gaussian with

$$\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\theta}_A \mathbf{x}, \alpha(\mathbf{x}) \boldsymbol{\theta}_B \text{diag}(\boldsymbol{\theta}_A^2 \mathbf{x}^2) \boldsymbol{\theta}_B^\top\right). \quad (112)$$

Proof. Let $\mathbf{z} = \boldsymbol{\omega}_A \mathbf{x} \in \mathbb{R}^r$, whose components are $z_i = \sum_{j=1}^d \omega_{A;ij} x_j$. Since the $\omega_{A;ij}$ are independent Gaussian random variables, each z_i is Gaussian, and thus $\mathbf{z}$ is Gaussian. The moments are computed as follows.

Mean. Using $\mathbb{E}[\omega_{A;ij}] = \theta_{A;ij}$,

$$\mathbb{E}[z_i] = \sum_{j=1}^d \mathbb{E}[\omega_{A;ij}] x_j = \sum_{j=1}^d \theta_{A;ij} x_j = (\boldsymbol{\theta}_A \mathbf{x})_i. \quad (113)$$

Hence,

$$\mathbb{E}[\mathbf{z}] = \boldsymbol{\theta}_A \mathbf{x}. \quad (114)$$

Covariance. For $i \neq k$, independence across rows implies

$$\text{Cov}(z_i, z_k) = 0. \quad (115)$$

For the variance of each component,

$$\text{Var}(z_i) = \sum_{j=1}^d \text{Var}(\omega_{A;ij}) x_j^2 = \sum_{j=1}^d \alpha(\mathbf{x}) \theta_{A;ij}^2 x_j^2. \quad (116)$$

Thus,

$$\text{Cov}(\mathbf{z}) = \alpha(\mathbf{x}) \text{diag}\left(\sum_{j=1}^d \theta_{A;ij}^2 x_j^2\right)_{i=1}^r = \alpha(\mathbf{x}) \text{diag}(\boldsymbol{\theta}_A^2 \mathbf{x}^2). \quad (117)$$

Distribution of $\mathbf{y}$. Finally,

$$\mathbf{y} = \boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \mathbf{z}, \quad (118)$$

which is an affine transformation of a Gaussian vector. Therefore,

$$\mathbb{E}[\mathbf{y}] = \boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\theta}_A \mathbf{x}, \quad (119)$$

$$\text{Cov}(\mathbf{y}) = \boldsymbol{\theta}_B \text{Cov}(\mathbf{z}) \boldsymbol{\theta}_B^\top. \quad (120)$$

Substituting $\text{Cov}(\mathbf{z})$ yields

$$\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\theta}_A \mathbf{x}, \alpha(\mathbf{x}) \boldsymbol{\theta}_B \text{diag}(\boldsymbol{\theta}_A^2 \mathbf{x}^2) \boldsymbol{\theta}_B^\top\right). \quad (121)$$

□

Given the Gaussian posterior predictive derived above, the remaining challenge is efficient sampling. Direct approaches require constructing and factorizing the full output covariance, which scales quadratically in the output dimension and is prohibitive in practice, as discussed in the main text. We next show how the low-rank structure of the LoRA parameterization can be exploited to obtain an exact yet computationally efficient sampling procedure.

Proposition 5.2 (Low-Rank Sampling). *Let the posterior predictive distribution of $\mathbf{y}$ be*

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\theta}_A \mathbf{x}, \Sigma), \quad (122)$$

with covariance

$$\Sigma = \alpha(\mathbf{x}) \boldsymbol{\theta}_B \text{diag}(\boldsymbol{\theta}_A^2 \mathbf{x}^2) \boldsymbol{\theta}_B^\top. \quad (123)$$

Define $\mathbf{d}(\mathbf{x}) = \alpha(\mathbf{x}) \boldsymbol{\theta}_A^2 \mathbf{x}^2 \in \mathbb{R}^r$. Then an exact sample from this distribution can be obtained as

$$\mathbf{y} = \boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\theta}_A \mathbf{x} + \boldsymbol{\theta}_B (\sqrt{\mathbf{d}(\mathbf{x})} \odot \boldsymbol{\epsilon}_d), \quad \boldsymbol{\epsilon}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r). \quad (124)$$

Proof. Let $\boldsymbol{\epsilon}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r)$ and define

$$\tilde{\mathbf{z}} = \boldsymbol{\theta}_A \mathbf{x} + \sqrt{\mathbf{d}(\mathbf{x})} \odot \boldsymbol{\epsilon}_d. \quad (125)$$

Since $\boldsymbol{\epsilon}_d$ is standard Gaussian, $\tilde{\mathbf{z}}$ is Gaussian with

$$\mathbb{E}[\tilde{\mathbf{z}}] = \boldsymbol{\theta}_A \mathbf{x}, \quad (126)$$

$$\text{Cov}(\tilde{\mathbf{z}}) = \text{diag}(\mathbf{d}(\mathbf{x})). \quad (127)$$

Now define

$$\mathbf{y} = \boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \tilde{\mathbf{z}}. \quad (128)$$

Since $\mathbf{y}$ is an affine transformation of a Gaussian vector, it is Gaussian with mean

$$\mathbb{E}[\mathbf{y}] = \boldsymbol{\theta}_0 \mathbf{x} + \boldsymbol{\theta}_B \boldsymbol{\theta}_A \mathbf{x}, \quad (129)$$

and covariance

$$\text{Cov}(\mathbf{y}) = \boldsymbol{\theta}_B \text{Cov}(\tilde{\mathbf{z}}) \boldsymbol{\theta}_B^\top = \boldsymbol{\theta}_B \text{diag}(\mathbf{d}(\mathbf{x})) \boldsymbol{\theta}_B^\top. \quad (130)$$

Substituting $\mathbf{d}(\mathbf{x}) = \alpha(\mathbf{x}) \boldsymbol{\theta}_A^2 \mathbf{x}^2$ yields

$$\text{Cov}(\mathbf{y}) = \alpha(\mathbf{x}) \boldsymbol{\theta}_B \text{diag}(\boldsymbol{\theta}_A^2 \mathbf{x}^2) \boldsymbol{\theta}_B^\top = \Sigma. \quad (131)$$

Therefore, $\mathbf{y}$ has the desired distribution, and the proposed sampling procedure is exact. $\square$

Having derived an exact posterior predictive distribution and an efficient sampling procedure, we now consider two complementary inference modes arising from the same stochastic LoRA formulation. The first is a *deterministic mode*, which replaces the stochastic reduction matrix with its posterior mean. This corresponds to a maximum a posteriori (MAP)-style approximation and enables fast inference, as the resulting LoRA weights can be directly merged into the base model without requiring sampling. The second is a *sampling mode*, where stochastic forward passes are performed using the posterior predictive distribution, enabling estimation of predictive uncertainty via Monte Carlo sampling.

DETERMINISTIC MODE. In the deterministic setting, we replace the stochastic LoRA parameters with their posterior means. For each LoRA layer, the adapted weights are:

$$\theta_{\text{adapted}} = \theta_0 + \theta_B \theta_A. \quad (132)$$

This recovers the standard LoRA formulation, where the low-rank update $\theta_B \theta_A$ is deterministically added to the base weights θ_0 . This form allows the adapted weights to be precomputed and merged into the base model prior to inference, does not require the inference network α , and thus results in no additional computational overhead at test time.

SAMPLING MODE. In the sampling setting, we retain the stochasticity of the reduction matrix. For each LoRA layer, we use:

$$\omega_{\text{adapted}} = \theta_0 + \theta_B \omega_A, \quad (133)$$

where the output $\omega_{\text{adapted}} \mathbf{x}$ is sampled (using Low-Rank Sampling) from the learned predictive distribution. Performing multiple stochastic forward passes $s = 1, \dots, S$ yields a set of predictive outputs from which the predictive mean and variance can be estimated via Monte Carlo:

$$\hat{\mathbb{E}}[\mathbf{y}] = \frac{1}{S} \sum_{s=1}^S \mathbf{y}^{(s)}, \quad \hat{\sigma}^2 = \frac{1}{S} \sum_{s=1}^S (\mathbf{y}^{(s)} - \hat{\mathbb{E}}[\mathbf{y}])^2. \quad (134)$$

Moreover, the predictive uncertainty can be further decomposed into epistemic and aleatoric components via the law of total variance:

$$\text{Var}(\mathbf{y} \mid \mathbf{x}) = \underbrace{\text{Var}_{\omega}(\mathbb{E}[\mathbf{y} \mid \mathbf{x}, \omega])}_{\text{epistemic}} + \underbrace{\mathbb{E}_{\omega}[\text{Var}(\mathbf{y} \mid \mathbf{x}, \omega)]}_{\text{aleatoric}}. \quad (135)$$

In general, $\mathbf{x}$ denotes the model input (e.g., a token sequence for language models or a feature vector in classification settings), processed by a neural backbone. Stochasticity is introduced through the LoRA weights ω_A . The epistemic component captures uncertainty over the adapted parameters, while the aleatoric component reflects the intrinsic predictive uncertainty of the base model conditioned on fixed weights.

5.6.2 KL Divergence Proof

During training we optimise a variational lower bound to jointly maximise predictive performance and control model complexity. This is achieved by balancing the expected log-likelihood with a KL regulariser between the variational posterior and the prior. In this section we derive the latter.

Proposition 5.3 (Closed-Form KL Divergence). *Let the variational posterior $q(\omega_A \mid \mathbf{x})$ be as defined in eq. (100) and the prior $p(\omega_A)$ be as defined in eq. (103). Then the KL divergence admits the closed form*

$$D_{KL}[q(\omega_A \mid \mathbf{x}) \parallel p(\omega_A)] = \frac{1}{2} \left(\frac{(\alpha(\mathbf{x}) + 1)(1 + p)}{p} + \log \frac{p}{1 - p} - \log \alpha(\mathbf{x}) - 1 \right),$$

which depends only on α and the dropout (prior) rate p hyperparameter.

Proof. Since both $q(\omega_A \mid \mathbf{x})$ and $p(\omega_A)$ factorise over entries, the KL divergence decomposes as

$$D_{KL}[q(\omega_A \mid \mathbf{x}) \parallel p(\omega_A)] = \sum_{i=1}^r \sum_{j=1}^d D_{KL}[q(\omega_{A;ij} \mid \mathbf{x}) \parallel p(\omega_{A;ij})]. \quad (136)$$

From eq. (100) and eq. (103), we identify

$$q(\omega_{A;ij} \mid \mathbf{x}) = \mathcal{N}(\theta_{A;ij}, \alpha(\mathbf{x})\theta_{A;ij}^2), \quad p(\omega_{A;ij}) = \mathcal{N}(0, \frac{p}{1-p}\theta_{A;ij}^2).$$

Applying the Gaussian KL formula and substituting $\sigma_q^2 = \alpha(\mathbf{x})\theta_{A;ij}^2$ and $\sigma_p^2 = \frac{p}{1-p}\theta_{A;ij}^2$, the factor $\theta_{A;ij}^2$ cancels in all terms, yielding

$$D_{\text{KL}}[q(\boldsymbol{\omega}_{A;ij} \mid \mathbf{x}) \parallel p(\boldsymbol{\omega}_{A;ij})] = \frac{1}{2} \left(\frac{\alpha(\mathbf{x}) + 1}{\frac{p}{1-p}} - 1 + \log \frac{p}{1-p} - \log \alpha(\mathbf{x}) \right). \quad (137)$$

Summing over all (i, j) gives the final expression in eq. (104). $\square$

In practice, we employ a *normalized* version of this KL term by dividing by the number of parameters rd and the KL maximum value, so that the regularisation strength is independent of the network width and the maximum KL value equals one.

5.7 EXPERIMENTAL DETAILS AND HYPERPARAMETERS

This appendix presents the dataset, metric, and architecture hyperparameter details for the experiments reported in the main text. All models were trained on 94 GB H100 Nvidia GPUs. Models are trained on the training split and evaluated on the test split. The validation split (when available) is used to select the best checkpoint; otherwise, the final training checkpoint is used.

5.7.1 Commonsense Reasoning

The commonsense reasoning task covers 8 sub-tasks (BoolQ, PIQA, SIQA, ARC-c, ARC-e, OBQA, HellaS, and WinoG), each with its own predefined train/test split. The objective is to correctly answer a given multiple-choice question. The models are trained and evaluated as sequence-based models.

DATASET GENERATION AND AVAILABILITY. We adopt the benchmark protocol of Z. Hu et al. (2023), directly using their released dataset construction and evaluation setup without modification. In particular, we use their Commonsense170K fine-tuning data, constructed by formatting the training sets of BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA using predefined task-specific templates. Each dataset is treated as a separate commonsense reasoning task, and all examples are converted into a unified

structured prompt consisting of a task description followed by the input context and multiple-choice options. The datasets and prompt templates are publicly available at: <https://github.com/AGI-Edgerunners/LLM-Adapters>.

METRICS. The primary evaluation metric is accuracy, defined as the proportion of correctly answered queries over the total number of queries in the evaluation set. In practice, model predictions are obtained via autoregressive generation with nucleus and beam decoding. Specifically, we use temperature sampling with temperature 0.1, top- p sampling with $p = 0.75$, top- k filtering with $k = 40$, and beam search with 4 beams, following Q. Huang et al. (2025). The model generates up to 32 new tokens conditioned on the input prompt. The generated sequence is post-processed by extracting the model’s response segment and parsing the first occurrence of a task-specific keyword using a deterministic rule-based function. This extracted token is treated as the predicted label $\hat{y}_i$. If no valid keyword is found, the prediction is marked as incorrect. Accuracy is computed over the full evaluation set as:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\hat{y}_i = y_i], \quad (138)$$

where N is the number of samples, y_i is the ground-truth label, and $\mathbb{I}[\cdot]$ is the indicator function.

ARCHITECTURE AND HYPERPARAMETERS. We adopt the same experimental setup as Q. Huang et al. (2025), training for 3 epochs using the AdamW optimizer (Loshchilov and Hutter 2017) with a linear learning-rate scheduler and a warmup of 100 gradient steps, an effective batch size of 16, a learning rate of 1×10^{-4} for Llama 3 and 2×10^{-4} for Llama 2, and no weight decay. The pretrained backbone is loaded in 8-bit quantization to reduce memory consumption while preserving inference quality, and training is performed in mixed precision (bf16). Following Q. Huang et al. (2025), we apply LoRA adapters to the attention and MLP projection matrices using rank 32 and LoRA alpha of 64.

The BaLoRA inference network predicts layer-wise uncertainty coefficients conditioned on the current input sequence. It consists of a frozen transformer encoder (i.e., the pretrained backbone, either Llama 2 or 3) used solely for

feature extraction, followed by a lightweight MLP that maps the final hidden representation of the last token to a set of coefficients over LoRA layers. The forward computation is:

$$\mathbf{x} \rightarrow \text{Frozen Transformer} \rightarrow h_{\text{last}} \rightarrow \text{MLP}(d \rightarrow 256 \rightarrow 256 \rightarrow L) \rightarrow \text{Softplus} \rightarrow \boldsymbol{\alpha}, \quad (139)$$

where L denotes the number of LoRA layers and $\boldsymbol{\alpha} \in \mathbb{R}^L$ are the predicted coefficients. The transformer encoder is kept entirely frozen and serves only as a feature extractor, while all trainable parameters reside in the MLP.

5.7.2 Visual Perception

The visual perception task covers 4 image classification benchmarks (CIFAR-10, CIFAR-100, Oxford-IIIT Pets, and Oxford Flowers-102), each with its own standard train/test split. The objective is to correctly classify a given input image into one of the predefined categories. The models are fine-tuned and evaluated using a Vision Transformer backbone with a linear classification head.

DATASET GENERATION AND AVAILABILITY. We evaluate BaLoRA on four standard image classification benchmarks: CIFAR-10, CIFAR-100, Oxford-IIIT Pets, and Oxford Flowers-102. We follow the standard dataset splits provided by each benchmark, using the official training and test partitions without modification. All models are fine-tuned on images resized to 224×224 and normalized using ImageNet mean and standard deviation. Data augmentation during training includes random horizontal flipping, random rotation, and color jittering, while evaluation is performed without augmentation.

METRICS. We report Top-1 classification accuracy and Expected Calibration Error (ECE) as evaluation metrics. Accuracy is computed as the fraction of correctly classified samples over the full test set:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\hat{y}_i = y_i], \quad (140)$$

where N is the number of test samples, y_i is the ground-truth label, $\hat{y}_i$ is the predicted label, and $\mathbb{I}[\cdot]$ is the indicator function.

ECE is computed using 15 bins with L1 normalization. Let B_m denote the set of indices in bin m , $\text{acc}(B_m)$ the empirical accuracy in that bin, and $\text{conf}(B_m)$ the average predicted confidence. Then:

$$\text{ECE} = \sum_{m=1}^{15} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|. \quad (141)$$

ARCHITECTURE AND HYPERPARAMETERS. We optimize using AdamW with learning rate 10^{-3} , no weight decay, batch size 512, and linear scheduling with 10% warmup. Gradient clipping is applied with a maximum norm of 1.0. All models are trained in mixed precision (bf16) using 8-bit loading for the backbone encoder. We follow the PEFT protocol of Dosovitskiy (2020) and apply LoRA adapters to the query and value projection matrices of all transformer attention layers, as implemented in the `timm` library. We use rank $r \in \{8, 16\}$ and scaling factor $\alpha \in \{16, 32\}$ depending on dataset complexity, with no additional modifications to the pretrained backbone.

The backbone model is a Vision Transformer (ViT-L/16) pretrained on ImageNet-21K (Dosovitskiy 2020). The model processes images as sequences of 16×16 patches and produces a sequence of token embeddings of dimension d . A classification token (CLS) is prepended to the patch sequence and used for downstream prediction.

The BaLoRA inference network employs a second Vision Transformer encoder as a frozen feature extractor. This encoder shares the same architectural family as the backbone (ViT), but is instantiated independently and loaded in 8-bit precision for memory efficiency. Unlike the backbone, this encoder is never updated and serves only to extract a global image representation from the CLS token. We ablate several encoder choices for this module (see Figure 24) and select DeiT-Base (Touvron, Cord, et al. 2021), distilled on ImageNet-1k, as it performs best on the validation set. Given an input image $\mathbf{x}$, the inference network computes:

$$\mathbf{x} \rightarrow \text{Frozen ViT Encoder} \rightarrow h_{\text{CLS}} \in \mathbb{R}^d \rightarrow \text{MLP}(d \rightarrow 256 \rightarrow 256 \rightarrow L) \rightarrow \text{Softplus} \rightarrow \alpha$$

where L is the number of LoRA layers. The ViT encoder is kept entirely frozen and serves only as a feature extractor, while all trainable parameters reside in the MLP.

5.7.3 Materials Property Prediction

The materials property prediction task targets bandgap regression on the QMOF database. The objective is to predict the DFT-computed electronic bandgap (in eV) of a given metal-organic framework structure. The models are fine-tuned and evaluated using a multimodal MOFTransformer backbone that jointly processes crystal graph and three-dimensional energy grid representations.

DATASET GENERATION AND AVAILABILITY. We evaluate BaLoRA on the QMOF bandgap prediction task using the QMOF database (Rosen, Iyer, et al. 2021), a standard regression benchmark for metal-organic framework (MOF) property prediction. The dataset consists of DFT-computed electronic bandgap values (in eV) for crystalline MOF structures. We follow the standard data splits provided by the benchmark without modification. The dataset is publicly available through the `moftransformer` package (Kang et al. 2023).

METRICS. The primary evaluation metric is Mean Absolute Error (MAE), computed between model predictions and ground-truth bandgap values on the test set:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i|, \quad (142)$$

where N is the number of test samples, y_i is the ground-truth bandgap, and $\hat{y}_i$ is the predicted value. Predictions are denormalized using the training-set mean $\mu = 2.0899$ and standard deviation $\sigma = 1.1295$ before computing the metric. For uncertainty estimation, we additionally assess calibration quality by performing $T = 100$ stochastic forward passes at test time, computing per-sample predictive variance, and reporting the Spearman rank correlation between predictive variance and squared prediction error. A higher correlation

indicates that the model’s uncertainty estimates are better aligned with its actual errors, reflecting more reliable uncertainty quantification.

ARCHITECTURE AND HYPERPARAMETERS. The backbone model is MOF-Transformer (Kang et al. 2023), a multimodal transformer that jointly processes crystal graph and three-dimensional energy grid representations of MOF structures. Following preliminary experiments, we apply LoRA adapters to the MLP projection matrices of all transformer blocks, using rank $r = 64$ and scaling factor $\alpha = 128$. Training is performed for 20 epochs using the AdamW optimizer (Loshchilov and Hutter 2017) with learning rate 5×10^{-4} , no weight decay, batch size 32, and a linear learning-rate schedule with 5% warmup. Gradient clipping is applied with a maximum norm of 1.0, and training uses mixed precision (bf16). For LoRA and DoRA baselines, a dropout rate of 0.1 is applied to the adapter weights; for BaLoRA, dropout is disabled. The regression head and all adapter weights are trained end-to-end using an L1 loss on normalized targets.

The BaLoRA inference network predicts layer-wise uncertainty coefficients conditioned on the input MOF structure. It reuses the frozen graph and volume embedding modules from the pretrained backbone as feature extractors. Given an input sample, graph token embeddings are obtained from the frozen CGCNN (Xie and Grossman 2018) encoder and aggregated via masked mean pooling over valid atom positions. The resulting graph-level representation is summed with the frozen volume embedding and passed through a lightweight MLP head:

$$\mathbf{x} \rightarrow \text{Frozen Graph Encoder} \rightarrow \bar{h}_{\text{graph}} \oplus h_{\text{vol}} \rightarrow \text{MLP}(d \rightarrow d/2 \rightarrow L) \rightarrow \text{Softplus} \rightarrow \alpha$$

where L denotes the number of LoRA layers, $\oplus$ denotes element-wise addition, and $\alpha \in \mathbb{R}^L$ are the predicted coefficients. The BaLoRA KL divergence term uses a prior dropout probability of 0.1. The graph and volume embedding parameters are kept entirely frozen; all trainable parameters reside in the MLP head.

Table 21: Prior-probability ablation. Accuracy across commonsense reasoning benchmarks for different values of the prior probability p in the KL-divergence loss.

Model	Prior Probability (p)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG
Llama-2-7B	0.8	72.08	84.17	79.99	70.90	84.89	81.00	91.44	84.29
	0.6	72.69	84.93	80.45	72.10	85.56	85.40	91.23	83.66
	0.4	71.13	83.68	76.31	71.50	84.76	83.80	90.55	83.74
	0.2	73.00	83.30	80.09	68.86	83.16	82.40	90.69	82.08
Llama-3-8B	0.8	76.42	89.99	81.78	80.89	91.20	88.40	96.11	88.40
	0.6	75.66	89.77	81.99	80.89	90.66	86.20	95.90	87.61
	0.4	75.02	89.61	80.35	79.01	90.32	86.20	95.49	86.58
	0.2	74.98	88.74	80.09	79.95	90.28	85.60	95.62	86.50

5.8 ADDITIONAL RESULTS

This section presents additional ablation studies and convergence analyses that complement the main experimental results. We first examine the sensitivity of BaLoRA to the prior probability hyperparameter on commonsense reasoning, then ablate the choice of frozen encoder in the inference network for visual perception, and finally analyze the convergence of uncertainty estimates as a function of Monte Carlo steps on materials property prediction.

5.8.1 Additional Results on Commonsense Reasoning

Table 21 reports the effect of the prior probability p in the BaLoRA KL divergence loss (Equation eq. (104)) across all eight commonsense reasoning benchmarks. For both Llama-2-7B and Llama-3-8B, accuracy remains stable across a wide range of prior values ($p \in \{0.2, 0.4, 0.6, 0.8\}$), with no single setting consistently dominating across all tasks. On Llama-2-7B, the best average performance is obtained at $p = 0.6$, while on Llama-3-8B, $p = 0.8$ yields the highest scores on most benchmarks. Importantly, the spread between the best and worst prior settings is modest in all cases, confirming that BaLoRA is robust to this hyperparameter and does not require expensive tuning of p in practice.

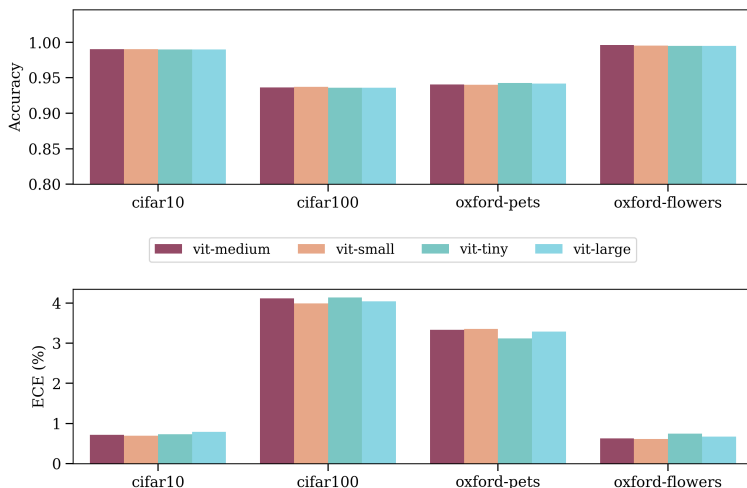


Figure 24: Inference-network encoder ablation. Accuracy (top) and ECE (bottom) for four Vision Transformer encoders: ViT-Large, ViT-Medium, ViT-Small, and ViT-Tiny.

5.8.2 Additional Results on Visual Perception

Figure 24 ablates the choice of frozen encoder used in the BaLoRA inference network for the visual perception experiments. We compare four Vision Transformer variants of increasing capacity: ViT-Tiny, ViT-Small, ViT-Medium (DeiT-Base), and ViT-Large (ViT-L/16). Across all four classification benchmarks, both accuracy (top) and ECE (bottom) are remarkably consistent regardless of encoder size. These results indicate that the BaLoRA inference network does not require a large or expensive encoder to produce well-calibrated uncertainty coefficients; even the smallest ViT-Tiny variant yields competitive performance, underscoring the lightweight nature of the approach.

5.8.3 Additional Results on Materials Property Prediction

Figure 25 shows the convergence behavior of BaLoRA on the QMOF bandgap prediction task as a function of the number of Monte Carlo (MC) forward passes at test time. The left panel reports MAE (in eV), which remains stable

at approximately 0.267 eV across all MC budgets from 10 to 100 steps, with narrow confidence bands indicating low variance across seeds. This confirms that the predictive mean is well-estimated with a modest number of stochastic passes, and additional MC steps do not degrade accuracy. The right panel reports the Spearman rank correlation between predictive variance and squared error, a measure of uncertainty calibration quality. A clear upward trend is observed: the correlation improves from approximately 0.315 at 10 MC steps to roughly 0.347 at 100 steps, with the most rapid gains occurring between 10 and 50 steps before plateauing. This indicates that while a small MC budget suffices for accurate point predictions, a moderately larger number of forward passes (≥ 50) is beneficial for obtaining well-ranked uncertainty estimates. The shaded confidence intervals widen slightly at higher MC budgets, reflecting increased seed-level variability in the correlation statistic, but the overall trend is monotonically increasing. Together, these results demonstrate that BaLoRA provides stable predictions and progressively improving uncertainty quantification at test time without any retraining.

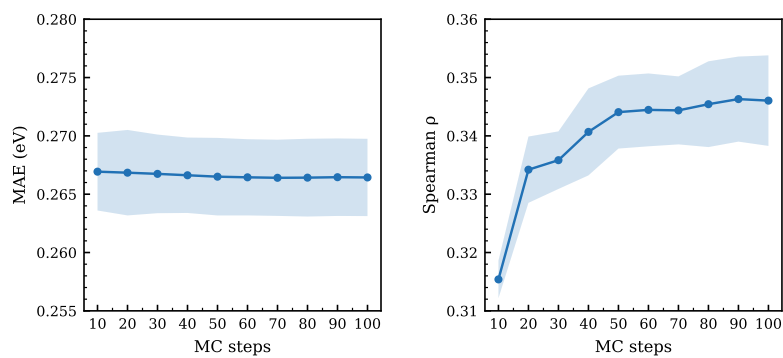


Figure 25: Monte Carlo convergence for bandgap prediction. MAE and Spearman correlation as the number of test-time stochastic forward passes increases. No retraining is performed; shaded regions show ± 1 standard deviation across three random seeds.

CONCLUSIONS

This thesis explores how principled uncertainty quantification can be integrated into scientific machine learning, with a focus on Bayesian methods that are efficient, scalable, and physically faithful. In particular, we set out to answer a central question: *Can we build scientific machine learning models that not only learn from data, but know what they do not know — producing calibrated uncertainty estimates efficiently and at scale?*

Having outlined the core contributions of this work in the preceding chapters, we are now in a position to address this overarching research question. Given the broad scope of scientific machine learning, spanning diverse physical modalities from dynamical systems to geometrically constrained atomistic structures, we break this central question into three research questions corresponding to three distinct classes of SciML models.

In Chapter 3, we addressed the question: *How can we equip sequence-based models with efficient, calibrated uncertainty estimates while preserving their structure?* To answer this, we introduced Bayesian Autoregressive and Recurrent Neural Network, a variational Bayesian framework that transforms autoregressive and recurrent models into Bayesian counterparts by jointly modelling network weights and observable states over time. By incorporating temporal structure directly into variational inference through a temporal lower bound and a time-dependent prior, BARNN produces uncertainty estimates that remain calibrated over sequential rollouts. Across applications in PDE modelling and molecular generation, we showed that this improves reliability in long-horizon prediction and better captures uncertainty accumulation in sequential processes.

In Chapter 4, we addressed the question: *How can we develop scalable Bayesian methods for SciML models defined on physically structured domains, such as atomistic systems, while remaining architecture-agnostic and computationally efficient?* To this end, we introduced BLIP, a variational Bayesian framework for message-passing neural networks that injects input-adaptive uncertainty into message and update functions while preserving physical symmetries. The framework is compatible with both invariant and equivariant architectures, enabling scalable uncertainty quantification across a broad class of atomistic models. Our results demonstrated improved predictive performance in data-scarce and out-of-distribution settings, together with uncertainty estimates that strongly correlate with model error and are directly useful for downstream tasks such as active learning.

Finally, in Chapter 5, we addressed the question: *How can we endow large pretrained models with calibrated uncertainty estimates during fine-tuning, without sacrificing accuracy or scalability?* To answer this, we introduced BaLoRA, a Bayesian extension of Low-Rank Adaptation in which the reduction matrix is treated probabilistically through input-adaptive noise injection. By performing posterior sampling directly in a low-dimensional latent space, BaLoRA preserves the computational efficiency of standard LoRA while improving predictive performance and uncertainty quality. Across reasoning, vision, and scientific prediction tasks, BaLoRA consistently improved over deterministic parameter-efficient fine-tuning and produced uncertainty estimates that correlated strongly with downstream error.

6.1 FUTURE DIRECTIONS

The frameworks developed in this thesis sit within a much broader effort to make Bayesian Deep Learning tractable for large-scale scientific models. It is worth situating them explicitly within this landscape before outlining where the work can go next.

At one end of the spectrum, Hamiltonian Monte Carlo remains the gold standard for posterior sampling but is computationally out of reach for models of the scale considered in this thesis (R. Zhang et al. 2019; Izmailov et al. 2021). Deep Ensembles (Lakshminarayanan et al. 2017) and their more efficient variants, such as Snapshot Ensembles (G. Huang et al. 2017) and SWAG (Maddox

et al. 2019), offer a practical middle ground, but their cost still scales linearly with ensemble size, which becomes prohibitive when models are embedded in iterative scientific pipelines such as molecular dynamics (Papamarkou et al. 2024). Last-layer Laplace approximations (Magnani, Krämer, et al. 2022; Magnani, Pförtner, et al. 2024) sit at the other extreme, offering a cheap, architecture-agnostic route to uncertainty that can be retrofitted onto any pre-trained model, but typically underperform ensembles out-of-distribution — precisely the regime that matters most in scientific applications. BARNN and BLIP were designed to occupy a different point in this trade-off space: a fully Bayesian treatment of network weights via a structured, input-adaptive variational distribution, aiming for the calibration of a full posterior treatment at a cost closer to that of a Laplace approximation. Encouragingly, this direction has continued to attract follow-up work since our contributions: PROBE (Mehdi et al. 2026) reframes MLIP uncertainty as a post-hoc selective-classification problem to avoid ensemble training entirely, while functionally-perturbed MLIPs (Zaghen et al. 2026) take an approach closer to BLIP in spirit, learning functional perturbations end-to-end with a proper scoring rule and, in some settings, surpassing it. That two independent lines of work have since pushed on exactly this trade-off suggests it is a genuinely open and active problem, not one this thesis closes.

Building on this landscape, several concrete directions follow from the contributions of this thesis.

Active learning. BLIP’s calibrated, error-correlated uncertainty estimates are directly suited to active learning for interatomic potentials (Kulichenko, Barros, et al. 2023; Jinnouchi et al. 2020; Zaverkin et al. 2024), where an acquisition function identifies configurations the model is least confident about and prioritises them for relabelling with expensive quantum-mechanical calculators. The same acquisition-function machinery (Ren et al. 2021; X. Wang et al. 2023) could be extended to BARNN-based PDE surrogates, where uncertainty could instead guide the selection of new parameter regimes or fidelity levels worth resolving at high cost (Musekamp et al. 2025a; Demo et al. 2023), and to BaLoRA-adapted foundation models, where uncertainty-guided fine-tuning could reduce the labelled data needed to adapt a pretrained model to a new scientific domain.

Transfer and continual learning. BaLoRA already shows that Bayesian parameter-efficient fine-tuning (E. J. Hu et al. 2022; N. Ding et al. 2023) can produce zero-shot uncertainty competitive with full ensembles. Treating the resulting posterior as a prior for subsequent tasks, rather than a final product, would give a principled route to continual adaptation as SciML foundation models (Choi et al. 2025) are repeatedly repurposed across downstream tasks with varying degrees of distribution shift.

Reinforcement learning and control. The Bayesian interpretation of reinforcement learning, in which priors encode beliefs over system dynamics and rewards act as likelihoods (Holdijk et al. 2023; Kappen and Ruiz 2016; Korbak et al. 2022), suggests a natural connection between uncertainty-aware inference and control. In this setting, the structured temporal priors developed for BARNN could provide a principled way to model uncertainty over evolving dynamics, enabling more robust decision-making in sequential scientific settings. Potential applications include steering molecular dynamics toward high-uncertainty regions for accelerated exploration (Oord et al. 2023) and improving uncertainty-aware control in data-driven turbulence modelling (Kurz, Offenhäuser, et al. 2023; Kurz, A. Beck, et al. 2025), where scalable Bayesian methods remain relatively underexplored.

Inference and compression at deployment scale. The $O(K \cdot C)$ cost of Monte Carlo uncertainty estimation remains a bottleneck when surrogates are queried millions of times within a single simulation (Coscia, Haan, et al. 2025), and many of the presented approaches ultimately still rely on repeated stochastic evaluation. Analytic moment propagation via local linearisation (R. Li et al. 2024) offers one direction for reducing this overhead. Indeed, recent work such as *Flowing with Confidence* (Kruiff et al. 2026b) demonstrates that, in continuous-time generative settings, uncertainty can be propagated through model dynamics using deterministic approximations of activation behaviour, substantially reducing or eliminating the need for Monte Carlo sampling while maintaining competitive estimation quality in those settings.

Beyond weight-space priors. BARNN’s time-dependent VAMP prior illustrates the value of encoding domain structure directly into the prior, rather than relying on generic weight-space assumptions. More broadly, functional priors informed by physical structure or simulation data (Meng et al. 2022; Psaros et al. 2023; Tran et al. 2022) suggest a richer alternative to standard

parameter-space constructions, where inductive bias is expressed at the level of functions rather than weights. This perspective highlights a broader design space for Bayesian SciML models in which prior structure is aligned more closely with the underlying data-generating process.

6.2 LIMITATIONS

Beyond the specific limitations of BARNN, BLIP, and BaLoRA, it is worth acknowledging where Bayesian methods as a class — including the ones developed here — fall short, and where alternative paradigms may be preferable.

All three frameworks rely on variational approximations rather than exact posterior inference. This is a deliberate trade-off for scalability, but it means the resulting posteriors can be overconfident or fail to capture multimodal structure that exact methods such as HMC would reveal (Izmailov et al. 2021; Garipov et al. 2018). The gap between our structured variational posteriors and the true posterior is, in general, unmeasured for the model scales considered here, since HMC itself is intractable at this scale.

Bayesian inference, even in these lightweight forms, is not free: BLIP requires roughly twice the FLOPs of standard training, and both BARNN and BaLoRA incur some overhead from noise injection and, when evaluating the predictive distribution, repeated sampling. Where computational budgets are tight or downstream tasks do not require calibrated uncertainty, this overhead may not be justified, and deterministic training or post-hoc calibration may be the more practical choice.

The quality of every result in this thesis is also bound by prior specification. We designed structured, data- or time-informed priors where possible, but specifying meaningful priors in high-dimensional weight space remains difficult in general (Papamarkou et al. 2024; Nalisnick 2018; Fortuin et al. 2021), and prior misspecification is a particular risk in the low-data, out-of-distribution regimes that motivate Bayesian SciML in the first place.

Finally, Bayesian inference is not the only principled framework for uncertainty quantification, and it is worth being explicit that it is not always the right tool. Conformal prediction (Fontana et al. 2023; Cairolì et al. 2023) provides distribution-free, finite-sample coverage guarantees without requiring any probabilistic model of the weights at all, sidestepping the prior-specification

and posterior-approximation issues above entirely. Cross-validation-based uncertainty estimates offer a similarly assumption-light alternative (Bishop and Nasrabadi 2006).

6.3 CLOSING REMARKS

This thesis set out to examine whether scientific machine learning models can be designed to recognise what they do not know, efficiently, at scale, and without sacrificing the physical structure that makes them useful in the first place. Across sequence-based models, geometrically constrained atomistic architectures, and large pretrained models adapted via fine-tuning, the results presented here suggest that this is indeed possible, and that it can be achieved without resorting to the two extremes that have historically dominated the field: prohibitively expensive full posterior sampling on one side, and cheap but comparatively uncalibrated approximations on the other.

More broadly, this work reflects a view consistent with the wider SciML literature surveyed throughout this thesis: uncertainty is most valuable not as an afterthought appended to a trained model, but as a first-class component of the modelling process. As SciML models continue to scale and are increasingly deployed in consequential settings such as materials discovery and climate modelling, it is unlikely that any single paradigm, whether Bayesian, conformal, or otherwise, will be sufficient on its own. What this thesis contributes is evidence that, for a substantial and practically important class of SciML models, principled Bayesian uncertainty is within reach at a cost the field can realistically afford, and a foundation on which the directions outlined above, and the parallel efforts of others already building on them, can continue to develop.

LIST OF PUBLICATIONS

The research presented in this thesis advances scalable Bayesian deep learning through three principal methodological contributions:

D. Coscia, M. Welling, N. Demo, and G. Rozza. *BARNN: A Bayesian Autoregressive and Recurrent Neural Network*. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, vol. 267, pp. 11379–11404, 2025.

D. Coscia, P. de Haan, and M. Welling. *BLIPs: Bayesian Learned Interatomic Potentials*. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. Also available as arXiv:2508.14022, 2025.

D. Coscia, S. Löwe, and M. Welling. *BaLoRA: Bayesian Low-Rank Adaptation of Large Scale Models*. arXiv preprint arXiv:2605.08110, 2026.

In addition to the work forming the core of this thesis, the author contributed to several related methods for uncertainty quantification. Although these publications complement the methodological themes developed here, they are not included as part of the thesis:

D. Coscia, and G. Rozza. *Bayesian Perspectives in Scientific Machine Learning: Methods, Challenges, and Opportunities*. In *Numerical Analysis and Scientific Computing: an Anthology*, Lecture Notes in Computational Science and Engineering, vol. 156, Springer, 2026.

F. de Kruiff, **D. Coscia**, M. Welling, and E. J. Bekkers. *Flowing with Confidence*. In arXiv preprint arXiv:2605.18472, 2026.

G. Canali, F. Olivo, **D. Coscia**, N. Demo, and G. Rozza. *Towards Uncertainty Quantification in Data-Driven Reduced-Order Models via Bayesian Graph Neural Networks*. In *AI&PDE: ICLR 2026 Workshop on AI and Partial Differential Equations*, 2026.

O. Zaghen, M. Zhdanov, **D. Coscia**, D. R. Wessels, and E. J. Bekkers. *Uncertainty-Aware Machine Learning Interatomic Potentials via Learned Functional Perturbations*. In arXiv preprint arXiv:2605.19939, 2026.

Finally, a substantial part of the author’s research activity was devoted to the design and development of PINA, an open-source Python library for physics-informed and scientific machine learning built within the PyTorch ecosystem:

D. Coscia, A. Ivagnes, N. Demo, and G. Rozza. *Physics-Informed Neural Networks for Advanced Modeling*. In *Journal of Open Source Software*, vol. 8, no. 87, article 5352, 2023.

PINA was also included in the PyTorch ecosystem in 2025.

DISCLOSURE ON THE USE OF LARGE LANGUAGE MODELS

Large language models were used solely to assist with language editing and polishing the presentation of this thesis. All scientific ideas, methodological developments, analyses, experiments, results, and conclusions presented in this work were conceived, conducted, and interpreted by the author under the supervision of the thesis supervisors.

BIBLIOGRAPHY

- Abramson, Josh, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. (2024). “Accurate structure prediction of biomolecular interactions with AlphaFold 3”. In: *Nature* 630.8016, pp. 493–500.
- Aghajanyan, Armen, Sonal Gupta, and Luke Zettlemoyer (2021). “Intrinsic dimensionality explains the effectiveness of language model fine-tuning”. In: *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 7319–7328.
- Aichberger, Lukas, Kajetan Schweighofer, and Sepp Hochreiter (2024). “Rethinking Uncertainty Estimation in Natural Language Generation”. In: *arXiv preprint arXiv:2412.15176*.
- Alemi, Alexander, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy (2018). “Fixing a Broken ELBO”. In: *International conference on machine learning*. PMLR, pp. 159–168.
- Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli (2020). “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *Advances in neural information processing systems* 33, pp. 12449–12460.
- Bar-Sinai, Yohai, Stephan Hoyer, Jason Hickey, and Michael P Brenner (2019). “Learning Data-Driven Discretizations for Partial Differential Equations”. In: *National Academy of Sciences* 116.31, pp. 15344–15349.
- Barroso-Luque, Luis, Muhammed Shuaibi, Xiang Fu, Brandon M Wood, Misko Dzamba, Meng Gao, Ammar Rizvi, C Lawrence Zitnick, and Zachary W Ulissi (2024). “Open Materials 2024 (OMat24) Inorganic Materials Dataset and Models”. In: *arXiv preprint arXiv:2410.12771*.
- Batatia, Ilyes, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi (2022). “MACE: Higher Order Equivariant Message Passing Neu-

- ral Networks for Fast and Accurate Force Fields". In: *Advances in neural information processing systems* 35, pp. 11423–11436.
- Bayes, Thomas (1763). "An Essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, A.M.F.R.S." In: *Philosophical transactions of the Royal Society of London*.
- Beck, Maximilian, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter (2024). "xlstm: Extended long short-term memory". In: *Advances in Neural Information Processing Systems* 37, pp. 107547–107603.
- Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). "A Neural Probabilistic Language Model". In: *Advances in neural information processing systems*.
- Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003). "A neural probabilistic language model". In: *Journal of machine learning research* 3, Feb, pp. 1137–1155.
- Bhattacharya, Kaushik, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart (2021). "Model Reduction and Neural Networks for Parametric PDEs". In: *The SMAI journal of computational mathematics* 7, pp. 121–157.
- Bishop, Christopher M and Nasser M Nasrabadi (2006). *Pattern recognition and machine learning*. Springer.
- Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518, pp. 859–877.
- Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Networks". In: *International conference on machine learning*. PMLR, pp. 1613–1622.
- Bodnar, Cristian, Wessel P Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A Weyn, Haiyu Dong, et al. (2025). "A foundation model for the Earth system". In: *Nature*, pp. 1–8.
- Born, Max and W Heisenberg (1985). "Zur quantentheorie der molekeln". In: *Original Scientific Papers Wissenschaftliche Originalarbeiten*. Springer, pp. 216–246.

- Bottou, Léon (2012). “Stochastic Gradient Descent Tricks”. In: *Neural Networks: Tricks of the Trade*. Springer, pp. 421–436.
- Brandstetter, Johannes, Max Welling, and Daniel E Worrall (2022). “Lie Point Symmetry Data Augmentation for Neural PDE Solvers”. In: *International Conference on Machine Learning*.
- Brandstetter, Johannes, Daniel E Worrall, and Max Welling (2022). “Message Passing Neural PDE Solvers”. In: *International Conference on Learning Representations*.
- Bronstein, Michael M, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst (2017). “Geometric deep learning: going beyond Euclidean data”. In: *IEEE Signal Processing Magazine* 34.4, pp. 18–42.
- Brooks, Steve, Andrew Gelman, Galin Jones, and Xiao-Li Meng (2011). *Handbook of markov chain monte carlo*. CRC press.
- Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33, pp. 1877–1901.
- Cairolì, Francesca, Nicola Paoletti, and Luca Bortolussi (2023). “Conformal quantitative predictive monitoring of STL requirements for stochastic processes”. In: *Proceedings of the 26th ACM international conference on hybrid systems: computation and control*, pp. 1–11.
- Cao, Zhonglin, Rishikesh Magar, Yuyang Wang, and Amir Barati Farimani (2023). “Moformer: self-supervised transformer model for metal–organic framework property prediction”. In: *Journal of the American Chemical Society* 145.5, pp. 2958–2967.
- Chen, Tianqi, Emily Fox, and Carlos Guestrin (2014). “Stochastic gradient hamiltonian monte carlo”. In: *International conference on machine learning*. PMLR, pp. 1683–1691.
- Chmiela, Stefan, Huziel E Sauceda, Igor Poltavsky, Klaus-Robert Müller, and Alexandre Tkatchenko (2019). “sGDML: Constructing accurate and data efficient molecular force fields using machine learning”. In: *Computer Physics Communications* 240, pp. 38–45.
- Choi, Youngsoo, Siu Wun Cheung, Youngkyu Kim, Ping-Hsuan Tsai, Alejandro N Diaz, Ivan Zanardi, Seung Whan Chung, Dylan Matthew Copeland, Coleman Kendrick, William Anderson, et al. (2025). “Defining Foundation

- Models for Computational Science: A Call for Clarity and Rigor". In: *arXiv preprint arXiv:2505.22904*.
- Cohen, Taco and Max Welling (2016). "Group equivariant convolutional networks". In: *International conference on machine learning*. PMLR, pp. 2990–2999.
- Cohen, Taco S, Mario Geiger, and Maurice Weiler (2019). "A General Theory of Equivariant CNNs on Homogeneous Spaces". In: *Advances in neural information processing systems* 32.
- Coscia, Dario, Nicola Demo, and Gianluigi Rozza (2024). "Generative Adversarial Reduced Order Modelling". In: *Scientific Reports*.
- Coscia, Dario, Pim de Haan, and Max Welling (2025). "BLIPs: Bayesian Learned Interatomic Potentials". In: *arXiv preprint arXiv:2508.14022*.
- Coscia, Dario, Anna Ivagnes, Nicola Demo, and Gianluigi Rozza (2023). "Physics-informed neural networks for advanced modeling". In: *Journal of Open Source Software* 8.87, p. 5352.
- Coscia, Dario, Max Welling, Nicola Demo, and Gianluigi Rozza (2025). "BARNN: A Bayesian Autoregressive and Recurrent Neural Network". In: *arXiv preprint arXiv:2501.18665*.
- Demo, Nicola, Marco Tezzele, and Gianluigi Rozza (2023). "A DeepONet multi-fidelity approach for residual learning in reduced order modeling". In: *Advanced Modeling and Simulation in Engineering Sciences* 10.1, p. 12.
- Depeweg, Stefan, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Stefan Udluft (2018). "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-Sensitive Learning". In: *International Conference on Machine Learning*.
- Dieleman, Sander, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. (2016). "Wavenet: A Generative Model for Raw Audio". In: *arXiv preprint arXiv:1609.03499*.
- Ding, Ning, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. (2023). "Parameter-efficient fine-tuning of large-scale pre-trained language models". In: *Nature machine intelligence* 5.3, pp. 220–235.
- Doob, Joseph L (1949). "Application of the theory of martingales". In: *Le calcul des probabilités et ses applications*, pp. 23–27.
- Dosovitskiy, Alexey (2020). "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929*.

- Eijkelboom, Floor, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent (2024). “Variational Flow Matching for Graph Generation”. In: *arXiv preprint arXiv:2406.04843*.
- Evans, Lawrence C (2022). *Partial differential equations*. Vol. 19. American mathematical society.
- Falcon, William and The PyTorch Lightning team (Mar. 2019). “PyTorch Lightning”. Version 1.4. In: *PyTorch Lightning*. DOI: [10.5281/zenodo.3828935](https://doi.org/10.5281/zenodo.3828935). URL: <https://github.com/Lightning-AI/lightning>.
- Fontana, Matteo, Gianluca Zeni, and Simone Vantini (2023). “Conformal prediction: a unified review of theory and new challenges”. In: *Bernoulli* 29.1, pp. 1–23.
- Fortuin, Vincent (2022). “Priors in Bayesian Deep Learning: A Review”. In: *International Statistical Review* 90.3, pp. 563–591.
- Fortuin, Vincent, Adrià Garriga-Alonso, Sebastian W Ober, Florian Wenzel, Gunnar Rätsch, Richard E Turner, Mark van der Wilk, and Laurence Aitchison (2021). “Bayesian neural network priors revisited”. In: *arXiv preprint arXiv:2102.06571*.
- Fu, Xiang, Brandon M Wood, Luis Barroso-Luque, Daniel S Levine, Meng Gao, Misko Dzamba, and C Lawrence Zitnick (2025). “Learning smooth and expressive interatomic potentials for physical property prediction”. In: *arXiv preprint arXiv:2502.12147*.
- Fuchs, Fabian, Daniel Worrall, Volker Fischer, and Max Welling (2020). “SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks”. In: *Advances in neural information processing systems* 33, pp. 1970–1981.
- Gal, Yarin and Zoubin Ghahramani (2016). “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *international conference on machine learning*. PMLR, pp. 1050–1059.
- Garipov, Timur, Pavel Izmailov, Dmitrii Podoprikin, Dmitry P Vetrov, and Andrew G Wilson (2018). “Loss surfaces, mode connectivity, and fast ensembling of dnns”. In: *Advances in neural information processing systems* 31.
- Gilmer, Justin, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl (2017). “Neural message passing for quantum chemistry”. In: *International conference on machine learning*. Pmlr, pp. 1263–1272.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.

- Grattafiori, Aaron, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. (2024). "The llama 3 herd of models". In: *arXiv preprint arXiv:2407.21783*.
- Graves, Alex (2011). "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 24.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International conference on learning representations*.
- Hinton, Geoffrey, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6, pp. 82–97.
- Hinton, Geoffrey E and Drew Van Camp (1993). "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights". In: *Conference on Computational Learning Theory*.
- Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*.
- Hoffman, Matthew D, David M Blei, Chong Wang, and John Paisley (2013). "Stochastic variational inference". In: *the Journal of machine Learning research* 14.1, pp. 1303–1347.
- Hohenberg, Pierre and Walter Kohn (1964). "Inhomogeneous electron gas". In: *Physical review* 136.3B, B864.
- Holdijk, Lars, Yuanqi Du, Ferry Hooft, Priyank Jaini, Berend Ensing, and Max Welling (2023). "Stochastic optimal control for collective variable free sampling of molecular transition paths". In: *Advances in Neural Information Processing Systems* 36, pp. 79540–79556.
- Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5, pp. 359–366.

- Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly (2019). "Parameter-efficient transfer learning for NLP". In: *International conference on machine learning*. PMLR, pp. 2790–2799.
- Hu, Edward J, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. (2022). "Lora: Low-rank adaptation of large language models." In: *ICLR 1.2*, p. 3.
- Hu, Yuge, Joseph Musielewicz, Zachary W Ulissi, and Andrew J Medford (2022). "Robust and scalable uncertainty estimation with conformal prediction for machine-learned interatomic potentials". In: *Machine Learning: Science and Technology* 3.4, p. 045028.
- Hu, Zhiqiang, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee (2023). "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models". In: *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5254–5276.
- Huang, Gao, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger (2017). "Snapshot ensembles: Train 1, get m for free". In: *arXiv preprint arXiv:1704.00109*.
- Huang, Qiushi, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang (2025). "HiRA: Parameter-efficient hadamard high-rank adaptation for large language models". In: *The Thirteenth International Conference on Learning Representations*.
- Izmailov, Pavel, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson (2021). "What are Bayesian neural network posteriors really like?" In: *International conference on machine learning*. PMLR, pp. 4629–4640.
- Jaegle, Andrew, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. (2021). "Perceiver io: A general architecture for structured inputs & outputs". In: *arXiv preprint arXiv:2107.14795*.
- Jiang, Ting, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, et al. (2024). "Mora: High-rank updating for parameter-efficient fine-tuning". In: *arXiv preprint arXiv:2405.12130*.

- Jiang, Zhengbao, Jun Araki, Haibo Ding, and Graham Neubig (2021). "How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering". In: *Transactions of the Association for Computational Linguistics*.
- Jinnouchi, Ryosuke, Kazutoshi Miwa, Ferenc Karsai, Georg Kresse, and Ryoji Asahi (2020). "On-the-fly active learning of interatomic potentials for large-scale atomistic simulations". In: *The Journal of Physical Chemistry Letters* 11.17, pp. 6946–6955.
- Jordan, Michael I, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul (1999). "An introduction to variational methods for graphical models". In: *Machine learning* 37.2, pp. 183–233.
- Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. (2021). "Highly accurate protein structure prediction with AlphaFold". In: *nature* 596.7873, pp. 583–589.
- Kang, Yeonghun, Hyunsoo Park, Berend Smit, and Jihan Kim (2023). "A multi-modal pre-training transformer for universal transfer learning in metal–organic frameworks". In: *Nature Machine Intelligence* 5.3, pp. 309–318.
- Kaplan, Aaron D, Runze Liu, Ji Qi, Tsz Wai Ko, Bowen Deng, Janosh Riebesell, Gerbrand Ceder, Kristin A Persson, and Shyue Ping Ong (2025). "A foundational potential energy surface dataset for materials". In: *arXiv preprint arXiv:2503.04070*.
- Kappen, Hilbert Johan and Hans Christian Ruiz (2016). "Adaptive importance sampling for control and inference". In: *Journal of Statistical Physics* 162.5, pp. 1244–1266.
- Kharitonov, Valery, Dmitry Molchanov, and Dmitry Vetrov (2018). "Variational Dropout via Empirical Bayes". In: *arXiv preprint arXiv:1811.00596*.
- Kingma, Diederik P (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.
- Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.
- Kingma, Durk P, Tim Salimans, and Max Welling (2015). "Variational Dropout and the Local Reparameterization Trick". In: *Advances in Neural Information Processing Systems*.

- Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.
- Kohl, Georg, Liwei Chen, and Nils Thuerey (2024). "Benchmarking Regressive Conditional Diffusion Models for Turbulent Flow Simulation". In: *ICML 2024 AI for Science Workshop*.
- Köhler, Jonas, Leon Klein, and Frank Noé (2019). "Equivariant Flows: sampling configurations for multi-body systems with symmetric energies". In: *arXiv preprint arXiv:1910.00753*.
- Kohn, Walter and Lu Jeu Sham (1965). "Self-consistent equations including exchange and correlation effects". In: *Physical review* 140.4A, A1133.
- Korbak, Tomasz, Ethan Perez, and Christopher Buckley (2022). "RL with KL penalties is better viewed as Bayesian inference". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1083–1091.
- Kreiman, Tobias and Aditi S Krishnapriyan (2025). "Understanding and Mitigating Distribution Shifts For Machine Learning Force Fields". In: *arXiv preprint arXiv:2503.08674*.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25.
- Kruijff, Friso de, Dario Coscia, Max Welling, and Erik Bekkers (2026a). *Flowing with Confidence*. arXiv: 2605.18472 [stat.ML]. URL: <https://arxiv.org/abs/2605.18472>.
- (2026b). "Flowing with Confidence". In: *arXiv preprint arXiv:2605.18472*.
- Kulichenko, Maksim, Kipton Barros, Nicholas Lubbers, Ying Wai Li, Richard Messerly, Sergei Tretiak, Justin S Smith, and Benjamin Nebgen (2023). "Uncertainty-driven dynamics for active learning of interatomic potentials". In: *Nature computational science* 3.3, pp. 230–239.
- Kulichenko, Maksim, Benjamin Nebgen, Nicholas Lubbers, Justin S Smith, Kipton Barros, Alice EA Allen, Adela Habib, Emily Shinkle, Nikita Fedik, Ying Wai Li, et al. (2024). "Data generation for machine learning interatomic potentials and beyond". In: *Chemical Reviews* 124.24, pp. 13681–13714.
- Kurz, Marius, Andrea Beck, and Benjamin Sanderse (2025). "Harnessing equivariance: Modeling turbulence with graph neural networks". In: *arXiv preprint arXiv:2504.07741*.

- Kurz, Marius, Philipp Offenhäuser, and Andrea Beck (2023). “Deep reinforcement learning for turbulence modeling in large eddy simulations”. In: *International journal of heat and fluid flow* 99, p. 109094.
- Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *Advances in neural information processing systems* 30.
- Lam, Remi, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. (2022). “GraphCast: Learning Skillful Medium-Range Global Weather Forecasting”. In: *arXiv preprint arXiv:2212.12794*.
- (2023). “Learning Skillful Medium-Range Global Weather Forecasting”. In: *American Association for the Advancement of Science*.
- Lester, Brian, Rami Al-Rfou, and Noah Constant (2021). “The power of scale for parameter-efficient prompt tuning”. In: *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059.
- Li, Rui, Marcus Klasson, Arno Solin, and Martin Trapp (2024). “Streamlining prediction in Bayesian deep learning”. In: *International Conference on Learning Representations*.
- Li, Yujia, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia (2018). “Learning Deep Generative Models of Graphs”. In: *arXiv preprint arXiv:1803.03324*.
- Li, Zongyi, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. (2020). “Fourier Neural Operator for Parametric Partial Differential Equations”. In: *International Conference on Learning Representations*.
- Li, Zongyi, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar (2024). “Physics-informed neural operator for learning partial differential equations”. In: *ACM/IMS Journal of Data Science* 1.3, pp. 1–27.
- Lippe, Phillip, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter (2024). “PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers”. In: *Advances in Neural Information Processing Systems*.
- Liu, Haotian, Chunyuan Li, Qingyang Wu, and Yong Jae Lee (2023). “Visual instruction tuning”. In: *International Conference on Neural Information Processing Systems* 36, pp. 34892–34916.

- Liu, Qi, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt (2018). “Constrained Graph Variational Autoencoders for Molecule Design”. In: *Advances in Neural Information Processing Systems* 31.
- Liu, Shih-Yang, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen (2024). “Dora: Weight-decomposed low-rank adaptation”. In: *Forty-first International Conference on Machine Learning*.
- Liu, Xiao, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang (2022). “P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68.
- Liu, Xiao, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang (2024). “GPT understands, too”. In: *AI open* 5, pp. 208–215.
- Loshchilov, Ilya and Frank Hutter (2017). “Decoupled Weight Decay Regularization”. In: *arXiv preprint arXiv:1711.05101*.
- Louizos, Christos and Max Welling (2016). “Structured and efficient variational deep learning with matrix gaussian posteriors”. In: *International conference on machine learning*. PMLR, pp. 1708–1716.
- (2017). “Multiplicative normalizing flows for variational bayesian neural networks”. In: *International conference on machine learning*. PMLR, pp. 2218–2227.
- Maddox, Wesley J, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson (2019). “A simple baseline for bayesian uncertainty in deep learning”. In: *Advances in neural information processing systems* 32.
- Magnani, Emilia, Nicholas Krämer, Runa Eschenhagen, Lorenzo Rosasco, and Philipp Hennig (2022). “Approximate Bayesian neural operators: Uncertainty quantification for parametric PDEs”. In: *arXiv preprint arXiv:2208.01565*.
- Magnani, Emilia, Marvin Pförtner, Tobias Weber, and Philipp Hennig (2024). “Linearization turns neural operators into function-valued Gaussian processes”. In: *arXiv preprint arXiv:2406.05072*.
- Mahabadi, Rabeeh Karimi, Sebastian Ruder, Mostafa Dehghani, and James Henderson (2021). “Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint*

- Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 565–576.
- Mehdi, Shams, Ilkwon Cho, and Olexandr Isayev (2026). “Knowing when to trust machine-learned interatomic potentials”. In: *arXiv preprint arXiv:2605.00640*.
- Meng, Xuhui, Liu Yang, Zhiping Mao, José del Águila Ferrandis, and George Em Karniadakis (2022). “Learning functional priors and posteriors from data and physics”. In: *Journal of Computational Physics* 457, p. 111073.
- Molchanov, Dmitry, Arsenii Ashukha, and Dmitry Vetrov (2017). “Variational Dropout Sparsifies Deep Neural Networks”. In: *International Conference on Machine Learning*.
- Musekamp, Daniel, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert (2025a). “Active Learning for Neural PDE Solvers”. In: *International Conference on Learning Representations*.
- (2025b). “Active learning for neural PDE solvers”. In: *International Conference on Learning Representations*. Vol. 2025, pp. 38777–38825.
- Nalisnick, Eric Thomas (2018). “On priors for Bayesian neural networks”. PhD thesis. University of California, Irvine.
- Nemeth, Christopher and Paul Fearnhead (2021). “Stochastic gradient markov chain monte carlo”. In: *Journal of the American Statistical Association* 116.533, pp. 433–450.
- Neumann, Mark, James Gin, Benjamin Rhodes, Steven Bennett, Zhiyi Li, Hitarth Choubisa, Arthur Hussey, and Jonathan Godwin (2024). “Orb: A fast, scalable neural network potential”. In: *arXiv preprint arXiv:2410.22570*.
- Oord, Cas van der, Matthias Sachs, Dávid Péter Kovács, Christoph Ortner, and Gábor Csányi (2023). “Hyperactive learning for data-driven interatomic potentials”. In: *npj Computational Materials* 9.1, p. 168.
- Özcelik, Rıza, Sarah de Ruiter, Emanuele Criscuolo, and Francesca Grisoni (2024). “Chemical Language Modeling with Structured State Space Sequence Models”. In: *Nature Communications* 15.1, p. 6176.
- Papamarkou, Theodore, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, et al. (2024). “Position: Bayesian Deep Learning is Needed in the Age of Large-Scale AI”. In: *International Conference on Machine Learning*. PMLR, pp. 39556–39586.

- Pastrana, Rafael, Deniz Oktay, Ryan P Adams, and Sigrid Adriaenssens (2023). "JAX FDM: A differentiable solver for inverse form-finding". In: *International Conference on Machine Learning*.
- Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. (2019). "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32.
- Pernot, Pascal (2022). "The long road to calibrated prediction uncertainty in computational chemistry". In: *The Journal of Chemical Physics* 156.11.
- Pfaff, Tobias, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia (2021). "Learning Mesh-Based Simulation with Graph Networks". In: *International Conference on Learning Representations*.
- Pichi, Federico, Beatriz Moya, and Jan S Hesthaven (2024). "A graph convolutional autoencoder approach to model order reduction for parametrized PDEs". In: *Journal of Computational Physics* 501, p. 112762.
- Price, Ilan, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. (2025). "Probabilistic weather forecasting with machine learning". In: *Nature* 637.8044, pp. 84–90.
- Psaros, Apostolos F, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis (2023). "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons". In: *Journal of Computational Physics* 477, p. 111902.
- Qin, Chengwei, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang (2023). "Is ChatGPT a general-purpose natural language processing task solver?" In: *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1339–1384.
- Quarteroni, Alfio, Paola Gervasio, and Francesco Regazzoni (2025). "Combining physics-based and data-driven models: advancing the frontiers of research with scientific machine learning". In: *arXiv preprint arXiv:2501.18708*.
- Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). "Improving Language Understanding by Generative Pre-Training". In: *OpenAI Blog*.
- Razdaibiedina, Anastasiia, Yuning Mao, Madian Khabza, Mike Lewis, Rui Hou, Jimmy Ba, and Amjad Almahairi (2023). "Residual prompt tuning:

- Improving prompt tuning with residual reparameterization". In: *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 6740–6757.
- Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.
- Ren, Pengzhen, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang (2021). "A survey of deep active learning". In: *ACM computing surveys (CSUR)* 54.9, pp. 1–40.
- Rezende, Danilo and Shakir Mohamed (2015). "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR, pp. 1530–1538.
- Rhodes, Benjamin, Sander Vandenhaute, Vaidotas Šimkus, James Gin, Jonathan Godwin, Tim Duignan, and Mark Neumann (2025). "Orb-v3: atomistic simulation at scale". In: *arXiv preprint arXiv:2504.06231*.
- Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *The annals of mathematical statistics*, pp. 400–407.
- Rosen, Andrew S, Victor Fung, Patrick Huck, Cody T O'Donnell, Matthew K Horton, Donald G Truhlar, Kristin A Persson, Justin M Notestein, and Randall Q Snurr (2022). "High-throughput predictions of metal–organic framework electronic properties: theoretical challenges, graph neural networks, and data exploration". In: *npj Computational Materials* 8.1, p. 112.
- Rosen, Andrew S, Shaelyn M Iyer, Debmalaya Ray, Zhenpeng Yao, Alan Aspuru-Guzik, Laura Gagliardi, Justin M Notestein, and Randall Q Snurr (2021). "Machine learning the quantum-chemical properties of metal–organic frameworks for accelerated materials discovery". In: *Matter* 4.5, pp. 1578–1597.
- Rozza, Gianluigi, Giovanni Stabile, and Francesco Ballarin (2022). *Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics*. SIAM.
- Rudner, Tim GJ, Zonghao Chen, Yee Whye Teh, and Yarin Gal (2022). "Tractable function-space variational inference in bayesian neural networks". In: *Advances in Neural Information Processing Systems* 35, pp. 22686–22698.
- Sanchez-Gonzalez, Alvaro, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia (2020). "Learning to Simulate Complex Physics with Graph Networks". In: *International Conference on Machine Learning*.

- Satorras, Victor Garcia, Emiel Hoogetboom, and Max Welling (2021). "E(n) Equivariant Graph Neural Networks". In: *International conference on machine learning*. PMLR, pp. 9323–9332.
- Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80.
- Schenck, Connor and Dieter Fox (2018). "Spnets: Differentiable fluid dynamics for deep neural networks". In: *Conference on Robot Learning*. PMLR, pp. 317–335.
- Schmidhuber, Jürgen, Sepp Hochreiter, et al. (1997). "Long Short-term Memory". In: *Neural Comput* 9.8, pp. 1735–1780.
- Schmidinger, Niklas, Lisa Schneckenreiter, Philipp Seidl, Johannes Schimunek, Pieter-Jan Hoedt, Johannes Brandstetter, Andreas Mayr, Sohvi Luukkonen, Sepp Hochreiter, and Günter Klambauer (2024). "Bio-xLSTM: Generative modeling, representation and in-context learning of biological and chemical sequences". In: *arXiv preprint arXiv:2411.04165*.
- Schütt, Kristof, Pieter-Jan Kindermans, Huziel Enoc Saucedo Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller (2017). "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in neural information processing systems* 30.
- Schütt, Kristof, Oliver Unke, and Michael Gastegger (2021). "Equivariant message passing for the prediction of tensorial properties and molecular spectra". In: *International conference on machine learning*. PMLR, pp. 9377–9388.
- Segler, Marwin HS, Thierry Kogej, Christian Tyrchan, and Mark P Waller (2018). "Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks". In: *ACS Central Science*.
- Shi, Chence, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang (2020). "GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation". In: *International Conference on Learning Representations*.
- Shoghi, Nima, Adeesh Kolluru, John R Kitchin, Zachary Ward Ulissi, C Lawrence Zitnick, and Brandon M Wood (2024). "From Molecules to Materials: Pre-training Large Generalizable Models for Atomic Property Prediction". In: *International Conference on Learning Representations*.

- Simm, Gregor, Robert Pinsler, and José Miguel Hernández-Lobato (2020). "Reinforcement Learning for Molecular Design Guided by Quantum Mechanics". In: *International Conference on Machine Learning*.
- Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.
- Tan, Aik Rui, Shingo Urata, Samuel Goldman, Johannes CB Dietschreit, and Rafael Gómez-Bombarelli (2023). "Single-model uncertainty quantification in neural network potentials does not consistently outperform model ensembles". In: *npj Computational Materials* 9.1, p. 225.
- Thaler, Stephan, Gregor Doehner, and Julija Zavadlav (2023). "Scalable Bayesian uncertainty quantification for neural network potentials: promise and pitfalls". In: *Journal of Chemical Theory and Computation* 19.14, pp. 4520–4532.
- Tomczak, Jakub and Max Welling (2018). "VAE with a VampPrior". In: *International Conference on Artificial Intelligence and Statistics*.
- Touvron, Hugo, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou (2021). "Training data-efficient image transformers & distillation through attention". In: *International conference on machine learning*. PMLR, pp. 10347–10357.
- Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. (2023). "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971*.
- Tran, Ba-Hien, Simone Rossi, Dimitrios Milios, and Maurizio Filippone (2022). "All you need is a good functional prior for Bayesian deep learning". In: *Journal of Machine Learning Research* 23.74, pp. 1–56.
- Unke, Oliver T, Stefan Chmiela, Huziel E Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T Schutt, Alexandre Tkatchenko, and Klaus-Robert Müller (2021). "Machine learning force fields". In: *Chemical Reviews* 121.16, pp. 10142–10186.
- Uria, Benigno, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle (2016). "Neural Autoregressive Distribution Estimation". In: *Journal of Machine Learning Research*.
- Van Baalen, Mart, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling (2020). "Bayesian Bits: Unifying

- Quantization and Pruning". In: *Advances in neural information processing systems*.
- Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.
- Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272. DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).
- Vladimirova, Mariia, Jakob Verbeek, Pablo Mesejo, and Julyan Arbel (2019). "Understanding priors in Bayesian neural networks at the unit level". In: *International Conference on Machine Learning*. PMLR, pp. 6458–6467.
- Wainwright, Martin J, Michael I Jordan, et al. (2008). "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2, pp. 1–305.
- Wang, Hanchen, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. (2023). "Scientific discovery in the age of artificial intelligence". In: *Nature* 620.7972, pp. 47–60.
- Wang, Xilu, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer (2023). "Recent advances in Bayesian optimization". In: *ACM Computing Surveys* 55.13s, pp. 1–36.
- Weiler, Maurice, Patrick Forré, Erik Verlinde, and Max Welling (2023). "Equivariant and Coordinate Independent Convolutional Networks". In: *A Gauge Field Theory of Neural Networks*, p. 110.

- Welling, Max and Yee W Teh (2011). "Bayesian Learning via Stochastic Gradient Langevin Dynamics". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688.
- Wilson, Andrew G and Pavel Izmailov (2020). "Bayesian deep learning and a probabilistic perspective of generalization". In: *Advances in neural information processing systems* 33, pp. 4697–4708.
- Wood, Brandon M, Misko Dzamba, Xiang Fu, Meng Gao, Muhammed Shuaibi, Luis Barroso-Luque, Kareem Abdelmaqsoud, Vahe Gharakhanyan, John R Kitchin, Daniel S Levine, et al. (2025). "UMA: A Family of Universal Models for Atoms". In: *arXiv preprint arXiv:2506.23971*.
- Xiao, Yuxin, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency (2022). "Uncertainty Quantification with Pre-trained Language Models: A Large-Scale Empirical Analysis". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*.
- Xie, Tian and Jeffrey C Grossman (2018). "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties". In: *Physical review letters* 120.14, p. 145301.
- Yang, Adam X, Maxime Robeyns, Xi Wang, and Laurence Aitchison (2022). "Bayesian Low-rank Adaptation for Large Language Models". In: *International Conference on Learning Representations*.
- Zaghen, Olga, Maksim Zhdanov, Dario Coscia, David R Wessels, and Erik J Bekkers (2026). "Uncertainty-aware Machine Learning Interatomic Potentials via Learned Functional Perturbations". In: *arXiv preprint arXiv:2605.19939*.
- Zaverkin, Viktor, David Holzmüller, Henrik Christiansen, Federico Errica, Francesco Alesiani, Makoto Takamoto, Mathias Niepert, and Johannes Kästner (2024). "Uncertainty-biased molecular dynamics for learning uniformly accurate interatomic potentials". In: *npj Computational Materials* 10.1, p. 83.
- Zdrazil, Barbara, Eloy Felix, Fiona Hunter, Emma J Manners, James Blackshaw, Sybilla Corbett, Marleen de Veij, Harris Ioannidis, David Mendez Lopez, Juan F Mosquera, et al. (2024). "The ChEMBL Database in 2023: a Drug Discovery Platform Spanning Multiple Bioactivity Data Types and Time Periods". In: *Nucleic Acids Research* 52, pp. D1180–D1192.
- Zhang, Ruqi, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson (2019). "Cyclical stochastic gradient MCMC for Bayesian deep learning". In: *arXiv preprint arXiv:1902.03932*.