Statistics and Applications {ISSN 2454-7395 (online)}
Special Issue on Recent Advances in Bayesian Statistics
and Machine Learning
Volume 24, No. 2, 2026 (New Series), pp 125–154
http://www.ssca.org.in/journal

Neural Architectures for Amortized Bayesian Inference: Statistical Foundations and Empirical Assessments

Roy Shivam Ram Shreshtth[1], **Arnab Hazra**[1] **and Gourab Mukherjee**[2]

[1]*Department of Statistics and Data Science,*
Indian Institute of Technology Kanpur, Kanpur, India 208016
[2]*Marshall School of Business,*
University of Southern California, Los Angeles, United States 90089-0809

Received: 13 January 2026; Revised: 8 May 2026; Accepted: 17 July 2026

Abstract

Since the turn of the century, approximate Bayesian inference has steadily evolved as new computational techniques have been incorporated to handle increasingly complex, large-scale predictive problems. The recent success of deep neural networks and foundation models has now given rise to a new paradigm in statistical modeling, in which Bayesian inference can be amortized through large-scale learned predictors. In amortized inference, substantial computation is required at the beginning to train a neural network, but it can subsequently produce approximate posteriors or predictions at much lower computational cost across a wide range of tasks. While the typical Bayesian inference procedures are computationally expensive due to repeated likelihood calculations and Monte Carlo steps for each new dataset, amortized inference provides a much lower computational cost at deployment.

Despite the growing popularity of amortized inference, its statistical interpretation and position within Bayesian inference remain poorly explored. In this paper, we present a statistical perspective on several major neural architectures, including feedforward networks, Deep Sets, and Transformers, and examine how they naturally support amortized Bayesian inference. We explore how these models perform structured approximation and also probabilistic reasoning in ways that yield controlled generalization error throughout a wide range of deployment scenarios, and how these properties can be harnessed for Bayesian computation. Via simulation studies, we evaluate the accuracy, robustness, and uncertainty quantification of amortized inference across varying sample sizes, varying noise distributional families, varying sparsity levels, and multimodality, highlighting its strengths and limitations.

Key words: Amortized inference; Deep sets; Feedforward neural network; Model misspecification; Multimodal posterior approximation; Transformer.

Corresponding Author: Arnab Hazra
Email: ahazra@iitk.ac.in

1. Introduction

Nearly twenty-five years ago, Breiman (2001) drew attention to a divide within statistical methodology between a data modeling culture, rooted in explicit probabilistic formulations, and an algorithmic modeling culture, which prioritizes predictive performance through flexible procedures that often lack a generative interpretation. While the former traditionally guided Bayesian analysis, the past two decades have seen a rapid expansion of high-dimensional and highly nonlinear models arising from machine learning, which increasingly influence Bayesian workflows and expand the set of problems for which fully probabilistic treatment is computationally feasible (Shmueli, 2010).

A major force behind this shift is the development of scalable Bayesian computation. Classical tools such as Markov chain Monte Carlo (MCMC) provide well-understood inferential guarantees (Gelfand and Smith, 1990). Still, they can become computationally prohibitive when applied repeatedly across large collections of related datasets or in models with expensive likelihoods. In many modern scientific pipelines, including climate modeling, astrophysics, genomics, and simulation-based inference, analysts must solve thousands or millions of Bayesian inverse problems within tight computational budgets. In such settings, the traditional per-instance paradigm of Bayesian computation becomes infeasible.

The concept of amortized Bayesian inference addresses this challenge by shifting the computational burden from inference at deployment time to a substantial, upfront training phase (Gershman and Goodman, 2014). Instead of running MCMC or optimizing a variational objective separately for each dataset, a flexible neural network is trained on a large collection of simulated or historical Bayesian inference tasks. Once trained, this network provides approximate posterior summaries, including point estimates, densities, or samples, for new datasets at negligible marginal cost. In effect, amortization constructs a global Bayesian surrogate, enabling rapid inference whenever new data arise from the same or a closely related generative mechanism. The benefits of amortization are well-illustrated by the training-deployment dynamics of modern large language models. For example, the BigScience Large Open-Science Open-access Multilingual Language Model (BLOOM) required more than a million GPU-hours on several hundred high-end processors during its four-month training phase, consuming an estimated 433.2 MWh of energy (Luccioni *et al.*, 2023). As pointed out by Zammit-Mangion *et al.* (2025), this amount of energy is roughly equivalent to the annual usage of 70 Australian households. Once this substantial upfront cost is incurred, the trained model can generate high-quality text on a single GPU at a negligible computational and energy cost. This stark disparity between the cost of training and the efficiency of subsequent inference epitomizes the principle of amortization: a large, one-time computational investment yields an inference mechanism that can be applied repeatedly and efficiently across new inputs.

Amortized Bayesian inference has recently emerged as a powerful alternative to likelihood-based inference for complex data structures, including complex spatial and extreme-value models, where exact likelihoods are intractable or prohibitively expensive (Sainsbury-Dale *et al.*, 2024; Zammit-Mangion *et al.*, 2025). In spatial statistics, Sainsbury-Dale *et al.* (2025) have shown that neural Bayes estimators and graph neural-network–based amortized inferences can deliver fast, accurate inference for Gaussian and non-Gaussian spatial fields,

irregularly spaced data, and high-dimensional latent structures, dramatically reducing computation relative to classical Monte Carlo or Integrated Nested Laplace Approximation (Rue *et al.*, 2009) approaches. In extreme-value analysis, where censored peaks-over-threshold models, heavy-tailed dependence, and spatiotemporal extremes yield challenging likelihood surfaces, amortized Bayesian inference has been employed to construct likelihood-free estimators, accelerate GEV-type Bayesian workflows, and facilitate inference under censoring and weak identifiability (Richards *et al.*, 2024). Collectively, these developments demonstrate that amortized Bayesian inference provides a scalable and flexible framework for tackling the computational bottlenecks inherent in modern spatial and extreme-value modeling, offering near-instant, reusable inference once a simulator-trained network has been fit.

Neural amortization strategies can be broadly classified according to the type of Bayesian quantity they approximate. The first regime, direct neural point estimation, learns a deterministic mapping from data to parameter estimates. Although limited to first-order summaries, this approach is useful in likelihood-free or simulation-based settings where exact Bayesian computation is challenging. Architectures such as Deep Sets (Zaheer *et al.*, 2017) and Set Transformers (Lee *et al.*, 2019) provide principled methods for encoding symmetries inherent in the likelihood or prior structure. Neural posterior estimation (NPE) extends this idea by learning conditional density estimators for the posterior (Greenberg *et al.*, 2019). The second regime, amortized variational inference (AVI), provides approximate posterior distributions rather than point estimates. These methods train neural networks to output parameters of an approximate posterior, effectively learning a variational family indexed by the data. Modern formulations often employ normalizing flows (Rezende and Mohamed, 2015; Papamakarios *et al.*, 2021) to represent posteriors with complex curvature, skewness, and multimodality that cannot be captured by mean-field approximations (Radev *et al.*, 2024). In Bayesian computation, AVI can be viewed as a simultaneous variational approximation over an entire distribution of tasks, rather than a single dataset. The third regime, neural samplers and transport methods, aims to accelerate Bayesian sampling by learning transformations that map complex posterior distributions to simpler latent distributions, where standard Monte Carlo procedures can mix rapidly. Neural transport maps (Hoffman *et al.*, 2019) and flow-matching methods (Lipman *et al.*, 2023) exemplify this approach. These techniques serve as learned reparameterizations or preconditioners for MCMC, yielding substantial computational savings for models with strongly correlated parameters or irregular posterior geometries.

The purpose of this paper is two-fold. We begin by reviewing the statistical background of a few machine learning tools and concepts specific to amortized Bayesian inference. Subsequently, we assess these amortized approaches from a Bayesian perspective, with particular attention to their reliability, structural limitations, and empirical performance relative to classical computational methods. While amortization can offer significant gains in speed when the training distribution accurately represents the deployment environment, its accuracy and stability degrade once the generative mechanism deviates from the conditions encountered during training. To study this behavior systematically, we conduct extensive simulation experiments along four axes. First, we investigate training heterogeneity, examining how variations in task size and diversity influence posterior accuracy and calibration. Second, we study data-scarce deployment and quantify the dependence of amortized posteriors on effective sample size. Third, we evaluate robustness under noise distributional

mismatch, characterizing how predictive and posterior accuracy deteriorate as covariates or structural aspects of the data-generating process drift from the training regime. Fourth, we examine whether amortized neural inference can accurately capture posterior distributions with disconnected support and complex topology, where conventional unimodal approximations are inadequate.

The paper proceeds as follows. Section 2 reviews the neural architectures employed: Feedforward Networks, Deep Sets, and Transformers, and clarifies their connection to Bayesian computation through kernel, variational, and transport-theoretic interpretations. Different aspects of amortized inference are summarized in Section 3. Section 4 presents simulation studies examining the four regimes described above under varying sample sizes, varying noise distributional families, varying sparsity levels, and multimodality.[1] Section 5 summarizes the findings and highlights some future research directions.

2. Popular neural network architectures

In this section, we discuss modern neural network architectures commonly used in amortized inference pipelines. By drawing analogies to classical statistical methods, we explain why each architecture is suited to particular inference tasks rather than treating them as black-box function approximators.

Notation and conventions. We write vectors as columns by default; the only exception is Section 2.3, where the input matrix $X \in \mathbb{R}^{N \times d_{\text{model}}}$ stacks token embeddings as rows. Throughout the paper, θ denotes the trainable weights of the amortized estimator and $\vartheta \in \Theta$ the inferential target (a generic Bayesian parameter). The symbol σ is used for both the component-wise nonlinear activation in an FNN (Section 2.1.1) and standard deviations elsewhere–the meaning is clear from context. By a *token* we mean an indexed unit of an input set; for tabular regression, a token is one observation (x_n, y_n). An *epoch* is one pass over the meta-training set of tasks, and the per-task sample size N_t is always distinguished from the meta-training-set size T_{train}.

2.1. Feedforward neural networks

The Feedforward Neural Network (FNN), also known as a multilayer perceptron (MLP) (Rumelhart *et al.*, 1986), is the foundational architecture for deep learning. The Universal Approximation Theorem (recalled in Section 2.1.1) guarantees that FNNs can represent arbitrarily complex nonlinear mappings between inputs and outputs.

The mathematical setup of FNN is as follows. Let $L \in \mathbb{N}$ denote the network depth. We define the width of each layer as n_l, where $l \in \{0, \ldots, L\}$. Consistent with our notation, the input dimension is $n_0 = d_{\text{in}}$ and the output dimension is $n_L = d_{\text{out}}$. A (dense) feedforward neural network is a parametrized mapping $f_\theta : \mathbb{R}^{d_{\text{in}}} \longrightarrow \mathbb{R}^{d_{\text{out}}}$, where $\theta = \left\{(W^{[l]}, b^{[l]})\right\}_{l=1}^{L}$, constructed on a strictly layered, acyclic computational graph with full connectivity between consecutive layers. Given an input vector $x \in \mathbb{R}^{d_{\text{in}}}$, the network computes its output through

[1] All code used to generate the results is provided at `https://github.com/Royshivam18/Neural-Amortized-Inference` to ensure full reproducibility.

a sequence of recursive transformations. The computation begins by setting the initial activation equal to the input, $a^{[0]} = x$. For each hidden layer $l = 1, \ldots, L-1$, the network forms an affine transformation of the activations of the previous layer as

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}, \tag{1}$$

and then applies a nonlinear activation function $\sigma^{[l]}$ component-wise to obtain

$$a^{[l]} = \sigma^{[l]}(z^{[l]}). \tag{2}$$

Here, $W^{[l]} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b^{[l]} \in \mathbb{R}^{n_l}$ in (1) denote the weight matrix and bias vector for layer l, respectively. An affine transformation of the last hidden-layer activation produces the final output of the network:

$$f_\theta(x) = W^{[L]} a^{[L-1]} + b^{[L]}$$

typically without an additional nonlinearity.

The nonlinear activation functions $\sigma^{[l]}$ in (2) are essential. For instance, the Rectified Linear Unit (ReLU), defined as $\sigma(z) = \max(0, z)$, restricts the network to piecewise-linear functions. Without these nonlinearities, the composition collapses to a single affine map $f_\theta(x) = W_{\text{total}} x + b_{\text{total}}$, and expressivity is lost.

The complete parameter set θ resides in the space $\Theta = \prod_{l=1}^{L} (\mathbb{R}^{n_l \times n_{l-1}} \times \mathbb{R}^{n_l})$, and the total number of scalar parameters is given by $N_\theta = \sum_{l=1}^{L} n_l (n_{l-1} + 1)$. This hierarchical composition of functions allows the FNN to construct highly complex decision boundaries from simple linear operations. Figure 1 summarizes this structure schematically.

2.1.1. Universal approximation and adaptive basis functions

The utility of FNNs in statistical inference is rigorously grounded in the Universal Approximation Theorem. Seminal works by Cybenko (1989) and Hornik (1991) established that standard multilayer feedforward networks are universal approximators. Formally, let $C(K)$ denote the space of continuous functions on a compact set $K \subset \mathbb{R}^{d_{\text{in}}}$. The theorem states that the set of functions representable by a neural network with at least one hidden layer and a non-polynomial activation function σ is dense in $C(K)$ with respect to the supremum norm. That is, for any target function $g \in C(K)$ and error tolerance $\epsilon > 0$, there exists a parameter configuration θ such that

$$\sup_{x \in K} |g(x) - f_\theta(x)| < \epsilon.$$

While the theorem guarantees the existence of an approximating network, the geometric nature of the approximation is determined by the choice of activation function.

Modern architectures predominantly use ReLU activations. As shown by Montufar *et al.* (2014) and Arora *et al.* (2018), a ReLU network is a piecewise-affine estimator: it partitions the input domain into finitely many convex polytopes and is affine on each. Nonlinear targets are thus approximated by tiling them with flat facets, analogous to a polygonal mesh approximating a curved surface. In contrast, classical activation functions like the logistic

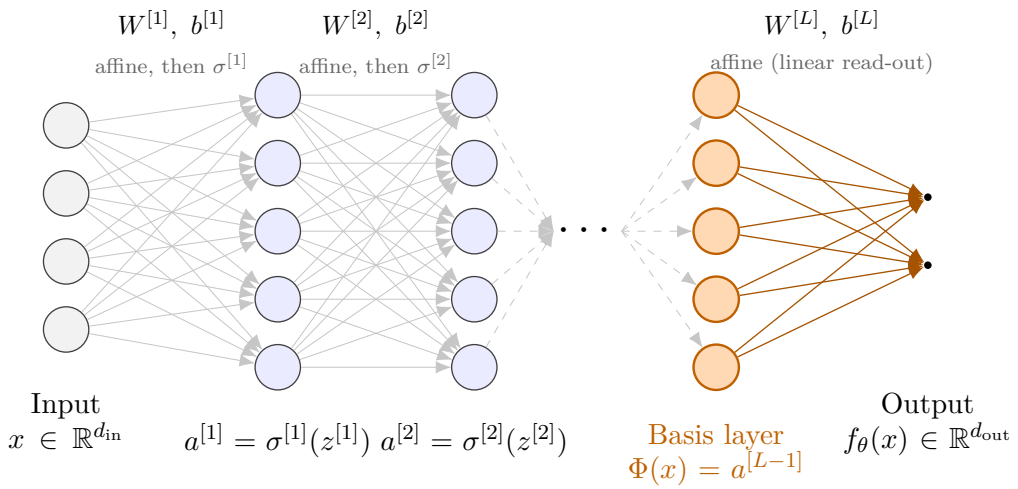


Figure 1: Architecture of an L-layer feedforward neural network. Each hidden layer applies an affine map $z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$ followed by the component-wise nonlinearity $\sigma^{[l]}$; the output layer is a linear read-out $f_\theta(x) = W^{[L]}a^{[L-1]} + b^{[L]}$. The penultimate layer $a^{[L-1]} = \Phi(x)$, shown in orange, is highlighted as a learned dictionary of *basis functions*: the network output is a linear combination of the entries of $\Phi(x)$, so the FNN realizes an adaptive basis expansion (Section 2.1.1). In the Transformer of Section 2.3, the query, key, and value projections are themselves FNN outputs of exactly this form, and the layer $\Phi(x)$ corresponds to the features on which attention then operates.

sigmoid, $\sigma(z) = (1 + e^{-z})^{-1}$, or the hyperbolic tangent, $\sigma(z) = \tanh(z)$, operate via smooth superposition. As established in the convergence rates derived by Barron (1993), networks using these smooth, bounded, sigmoidal functions approximate the target function by superimposing smooth 'steps' or 'ridges'. Unlike the sharp boundaries of ReLU polytopes, these activations provide a differentiable approximation where nonlinearity arises from the smooth saturation of neurons (Cybenko, 1989).

We can regard the first hidden layer as learning a dictionary of feature maps denoted by $\Phi(x) = (\phi_1(x), \ldots, \phi_{n_1}(x))^\top$, where each $\phi_i(x) := \sigma((W_i^{[1]})^\top x + b_i^{[1]})$ is a nonlinear basis function. The second hidden layer then computes weighted affine combinations of these features. Analytically, the network output can be written as a basis expansion given by

$$f_\theta(x) \;=\; c + \sum_{j=1}^{n_2} \beta_j \sigma\Big((W_j^{[2]})^\top \Phi(x) + b_j^{[2]}\Big),$$

where $W_j^{[2]}$ denotes the j-th row of the weight matrix $W^{[2]}$. This expression clarifies that the network is learning a new family of basis functions $\{\sigma((W_j^{[2]})^\top \Phi(x) + b_j^{[2]})\}_{j=1}^{n_2}$ built upon the first-layer dictionary, rather than using a fixed basis. Unlike classical nonparametric regression, where the basis (e.g., Fourier or spline) is fixed *a priori*, the FNN adaptively learns the basis parameters to represent the data structure best. This adaptive capability allows the FNN to emulate classical bases. Let $\mathcal{G} = \{g_1, \ldots, g_p\} \subset C(K)$ be any finite collection of continuous functions (a prescribed basis). By the universal approximation

theorem, for any $\epsilon > 0$, there exist parameters such that

$$\sup_{x \in K} \left| f_\theta(x) - \sum_{j=1}^{p} \beta_j g_j(x) \right| \; < \; \epsilon.$$

Equivalently, there exists a partition of the hidden units into blocks $\{\mathcal{I}_j\}_{j=1}^{p}$ such that each block realizes an approximation:

$$\tilde{g}_j(x) \; = \; \sum_{i \in \mathcal{I}_j} \alpha_i^{(j)} \sigma((W_i^{[1]})^\top x + b_i^{[1]}), \quad \sup_{x \in K} |g_j(x) - \tilde{g}_j(x)| < \varepsilon / p.$$

The second layer learns coefficients to form $\sum_j \beta_j \tilde{g}_j(x)$. Hence, even if the first layer (dictionary) is fixed, a two-hidden-layer network can approximate any finite linear combination of target basis functions on K.

This perspective allows us to map FNNs directly to classical kernel methods. To see this connection, we shift our focus from a trained deep network to a single-hidden-layer network with randomized first-layer parameters:

$$W_i^{[1]} \sim P_W, \qquad b_i^{[1]} \sim P_b.$$

If we freeze these parameters at initialization and only train the output-layer weights, each hidden unit activation $\phi_i(x) = \sigma((W_i^{[1]})^\top x + b_i^{[1]})$ acts strictly as a random feature (Rahimi and Recht, 2007). The induced kernel

$$k(x, x') \; := \; \mathbb{E}_{W,b} \left[\sigma((W^{[1]})^\top x + b^{[1]}) \sigma((W^{[1]})^\top x' + b^{[1]}) \right]$$

is well-defined under P_W and P_b. Training the output coefficients $\{\beta_j\}_{j=1}^{n_1}$ with squared loss and an ℓ_2-penalty is equivalent to kernel ridge regression with kernel $k(\cdot, \cdot)$. Moreover, Neal (1996) showed that if we place independent Gaussian priors on these output weights and let the hidden layer width $n_1 \to \infty$ with appropriate scaling, the induced prior over functions converges to a Gaussian Process (GP) with covariance $k(\cdot, \cdot)$. In this regime, the posterior mean predictor coincides with the GP regression estimator under kernel $k(\cdot, \cdot)$.

This basis-learning behavior is observable empirically. On algorithmic tasks such as modular addition ($a + b \mod P$), networks initially memorize the training data and then, after extended training, transition to perfect generalization–a phenomenon known as *grokking* (Power *et al.*, 2022). In the generalizing phase, the embedding layers converge to Fourier-like bases (sines and cosines) over the cyclic group (Nanda *et al.*, 2023), an empirical confirmation that the network discovers a problem-appropriate basis rather than fitting a fixed dictionary.

2.2. Deep sets as the summary statistic machine

Standard FNNs are inherently limited when applied to statistical inference tasks involving sets of independent and identically distributed (IID) observations. A standard MLP requires a fixed-size input vector, yet datasets in inference tasks naturally possess variable

cardinality. Furthermore, the joint probability of an IID set, $p(x_1, \ldots, x_N)$, is invariant to permutations of the indices. Consequently, any estimator $\widehat{\theta}(x_1, \ldots, x_N)$ derived from such data must respect this exchangeability. Standard architectures, which process inputs as ordered sequences, fail to capture this symmetry efficiently. To address this, Zaheer *et al.* (2017) proposed the 'Deep Sets' architecture, which provides a theoretically grounded framework for learning permutation-invariant functions.

Consider a set of inputs $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ where each element $x_j \in \mathbb{R}^{d_{\text{in}}}$. The Deep Sets architecture processes this set through a three-stage mechanism to ensure permutation invariance. First, an embedding function (encoder) $\phi : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{latent}}}$, typically parameterized as an MLP, maps each individual element to a latent representation:

$$h_j = \phi(x_j), \quad j = 1, \ldots, N.$$

Crucially, this mapping is applied identically to every element in the set, ensuring that feature extraction is independent of the position of the element.

Subsequently, a permutation-invariant aggregation operator is applied to the set of latent codes $\{h_1, \ldots, h_N\}$. While operators such as the mean or maximum are permissible, sum-pooling is often preferred in theoretical contexts as it preserves information regarding the set size N, which is vital for Bayesian inference. The aggregated summary statistic Z is computed as

$$Z = \sum_{j=1}^{N} h_j = \sum_{j=1}^{N} \phi(x_j).$$

Finally, a second neural network $\rho : \mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^{d_{\text{out}}}$, referred to as the decoder or regressor, processes the aggregated summary statistic to produce the final estimator

$$\widehat{y} = \rho(Z).$$

Combining these steps, the complete Deep Sets estimator can be expressed in the compact form

$$f(\mathcal{X}) \;=\; \rho\left(\sum_{x \in \mathcal{X}} \phi(x)\right). \tag{3}$$

Figure 2 depicts this three-stage construction.

From a statistical perspective, this architecture acts as a neural generalization of the Method of Moments (MoM). In classical estimation, parameters θ are estimated by equating empirical moments to theoretical moments. This estimation involves computing a vector of sample moments, $\hat{m}_k = \frac{1}{N}\sum x_j^k$, and applying an inverse mapping g to solve for θ. The Deep Sets formulation mirrors this structure exactly: the encoder $\phi(x)$ learns to extract optimal 'generalized moments' (sufficient statistics) rather than relying on fixed polynomial powers; the summation $\sum \phi(x_j)$ accumulates these statistics into a fixed-size summary; and the outer network ρ approximates the inverse mapping g.

This analogy extends to the Moment Generating Function (MGF). The Taylor expansion of the MGF, $M_X(t) = \mathbb{E}[e^{t^\top X}]$, encapsulates all moments of the distribution. If the

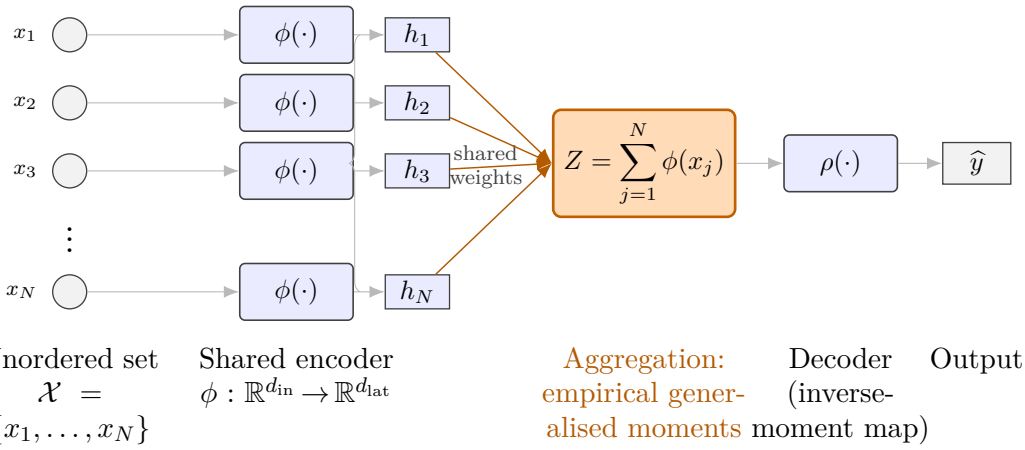


Figure 2: Deep Sets architecture. Each element x_j of an unordered set $\mathcal{X} = \{x_1, \ldots, x_N\}$ is passed through the same encoder ϕ (shared parameters), producing latent codes $h_j = \phi(x_j)$. The codes are aggregated by a permutation-invariant sum, $Z = \sum_{j=1}^{N} \phi(x_j)$, shown in orange. This aggregation block acts as a vector of *empirical generalized moments*: it is the neural analog of the classical method-of-moments statistic $\hat{m}_k = \frac{1}{N}\sum_j x_j^k$, except that the fixed polynomial basis is replaced by the learned features ϕ. The decoder ρ then approximates the inverse moment-to-parameter map and returns $\widehat{y} = \rho(Z)$.

encoder $\phi(x)$ possesses sufficient capacity, the aggregated vector $\sum \phi(x)$ effectively approximates the MGF of the underlying empirical distribution. Since the MGF uniquely characterizes the distribution under mild conditions, a Deep Set that learns a sufficient number of these moments captures a finite-dimensional representation of the entire probability measure (Wagstaff *et al.*, 2019). Theoretically, this principle is grounded in the Fisher-Neyman Factorization Theorem, which states that for Exponential Families, the likelihood factorizes through a sufficient statistic $T(x)$. Deep Sets effectively learn this statistic via ϕ, making them 'Neural Sufficient Statistic Learners'.

2.3. The transformer as a statistical operator

Statistically, a Transformer layer can be read as a learned, data-dependent kernel smoother on the input set. The variant we use in the experiments is the Set Transformer of Lee *et al.* (2019), which applies the same construction described below to an *unordered* input set (no positional encoding), yielding a permutation-equivariant operator over samples. The kernel-smoothing interpretation developed in this subsection applies identically to both.

Let $X \in \mathbb{R}^{N \times d_{\text{model}}}$ denote the input sequence of token embeddings, where the row $x_j \in \mathbb{R}^{d_{\text{model}}}$ corresponds to the j-th token. (We use the term *token* for an indexed unit of the input set; for tabular regression, a token is one observation (x_n, y_n).) Each Transformer layer (Vaswani *et al.*, 2017) projects every token through three learned feed-forward networks $\phi_Q, \phi_K, \phi_V : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}^{d_k}$, yielding queries, keys, and values:

$$q_j = \phi_Q(x_j), \quad k_j = \phi_K(x_j), \quad v_j = \phi_V(x_j),$$

where q_j, k_j, v_j form the j-th rows of $Q, K, V \in \mathbb{R}^{N \times d_k}$. In the original Transformer of

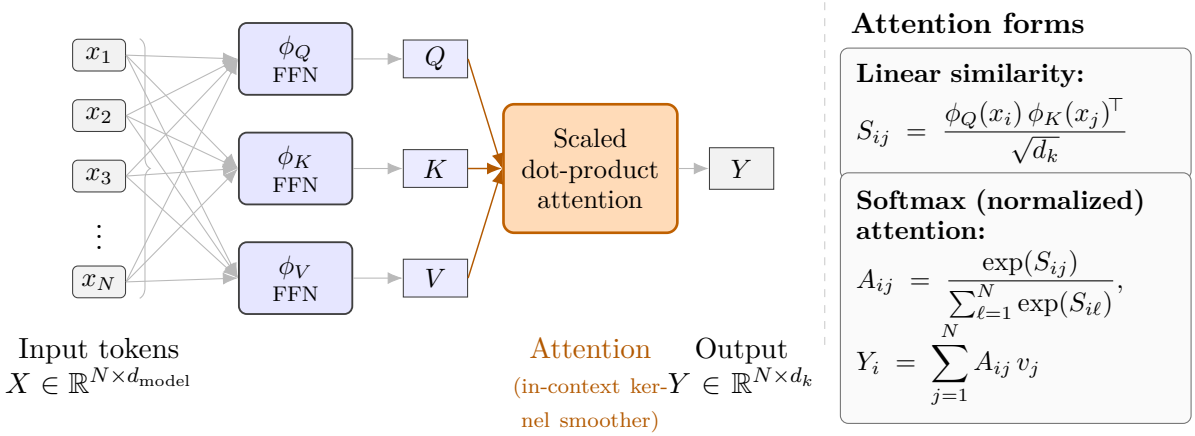


Figure 3: Transformer block. Every token x_j is fed through three independent feed-forward networks ϕ_Q, ϕ_K, ϕ_V–not bare matrix multiplications–producing per-token queries $q_j = \phi_Q(x_j)$, keys $k_j = \phi_K(x_j)$, and values $v_j = \phi_V(x_j)$. The scaled dot-product attention block (orange) aggregates value vectors using a data-dependent kernel induced by the query and key networks. The right panel lists the two attention strategies referenced in this section: the linear (unnormalized) similarity score S_{ij}, and the standard softmax-normalized attention A_{ij} used to mix the values into the output Y_i.

Vaswani *et al.* (2017), each of ϕ_Q, ϕ_K, ϕ_V is a single linear layer (*i.e.*, $\phi_Q(x_j) = x_j W_Q$ for a learned matrix W_Q, and similarly for K, V); modern variants frequently replace these with small multilayer FFNs to increase expressivity. The mechanism then computes a pairwise similarity matrix $S \in \mathbb{R}^{N \times N}$ with entries $S_{ij} = \phi_Q(x_i)\,\phi_K(x_j)^\top$, representing the unnormalized alignment between the i-th query and the j-th key. The scores are scaled by $1/\sqrt{d_k}$ and passed through a row-wise softmax to yield the attention matrix A,

$$A_{ij} = \frac{\exp(S_{ij}/\sqrt{d_k})}{\sum_{\ell=1}^{N} \exp(S_{i\ell}/\sqrt{d_k})}.$$

The layer output is $Y = AV$, *i.e.*, $Y_i = \sum_{j=1}^{N} A_{ij} v_j$. In standard implementations, several such *heads* (parallel attention computations with independent ϕ_Q, ϕ_K, ϕ_V) are concatenated to capture diverse dependencies. Figure 3 summarises this pipeline and lists the two attention forms commonly used downstream.

The attention matrix A admits a direct interpretation as a Nadaraya–Watson kernel regression estimator (Nadaraya, 1964; Watson, 1964). Define a data-dependent kernel κ_η parameterized by the projection networks $\eta = \{\phi_Q, \phi_K\}$ as

$$\kappa_\eta(x_i, x_j) = \exp\left(\frac{\phi_Q(x_i)\,\phi_K(x_j)^\top}{\sqrt{d_k}}\right).$$

Unlike stationary kernels used in nonparametric regression (*e.g.*, the Radial Basis Function; see Wand and Jones, 1994) that depend only on the Euclidean distance $\|x_i - x_j\|$, this kernel is non-stationary: it learns a metric in which similarity is determined by alignment of the projected features.

Under this definition, A_{ij} is precisely the normalised kernel weight, and Y_i is the Nadaraya–Watson estimator of the values V at the query location x_i, conditioned on the context $\{x_j\}$:

$$Y_i \;=\; \mathbb{E}_{\hat{P}}[V \mid x_i] \;=\; \sum_{j=1}^{N} \frac{\kappa_\eta(x_i, x_j)}{\sum_{\ell=1}^{N} \kappa_\eta(x_i, x_\ell)} v_j. \tag{4}$$

Statistically, the self-attention layer performs in-context kernel smoothing: it integrates information across the value vectors V, with the smoothing bandwidth and shape dictated by the learned similarity structure κ_η.

This kernel-smoothing interpretation invites a comparison with Gaussian Process (GP) regression. For a GP with kernel $k(\cdot,\cdot)$ and noise variance σ_n^2, the posterior mean at a test point x_* given training data $(X, \mathbf{y})$ is $m(x_*) = k(x_*, X)(K(X,X) + \sigma_n^2 I)^{-1}\mathbf{y}$. Both architectures form a weighted sum of targets, but their mixing mechanisms differ. The GP posterior requires inverting the kernel matrix, an $\mathcal{O}(N^3)$ operation; attention replaces this inversion with a row-wise softmax that scales as $\mathcal{O}(N^2)$. Softmax normalization is only an approximation to the inversion step, but it allows the Transformer to scale to substantially longer sequences while retaining the capacity to model complex dependencies (Rasmussen and Williams, 2006).

The query and key networks ϕ_Q, ϕ_K also admit a spectral interpretation via Mercer's theorem. A Mercer kernel $\kappa(x, x')$ admits the eigenfunction expansion $\sum_r \lambda_r \psi_r(x)\psi_r(x')$, and the Transformer approximates this with a finite-rank inner product of feature maps,

$$\phi_Q(x_i)\, \phi_K(x_j)^\top \;=\; \langle \phi_Q(x_i),\; \phi_K(x_j) \rangle,$$

where ϕ_Q and ϕ_K are learned feature maps. Standard attention decouples these projections ($\phi_Q \neq \phi_K$)– a choice motivated by causal or temporal asymmetry in language modeling–so the resulting attention matrix is not a Mercer kernel in the classical (symmetric) sense. Variants that tie the projections, such as the Reformer (Kitaev *et al.*, 2020), recover symmetry by setting $\phi_Q = \phi_K$ and can then exploit locality-sensitive hashing (Andoni and Indyk, 2008).

A full Transformer stack consists of multiple attention layers rather than a single one. This composition has a natural analog in Deep Gaussian Processes (DGPs; Damianou and Lawrence, 2013), where the output of one Gaussian process serves as the input to the next, warping the data geometry layer by layer. Each attention layer computes a data-dependent kernel over the features extracted by the previous layer, so the depth of the stack enables the model to represent strongly non-stationary dependencies that a single shallow kernel cannot capture.

3. Amortized Bayesian inference

Amortized Bayesian inference is a computational strategy in which a substantial initial computational budget is allocated to training a global model, thereby making subsequent inference for individual test cases computationally efficient. This approach is particularly potent when combined with deep learning, replacing complex, iterative computational steps (such as MCMC or numerical optimization) with efficient forward passes of a neural network.

In many high-dimensional settings, such as image analysis or large-scale regression, the raw data x often contains redundant information. Neural networks allow us to compress x into lower-dimensional summary statistics, which classical statistical techniques can then leverage. In this work, we specifically focus on two complementary regimes of amortized inference: point estimation and uncertainty quantification.

3.1. Point estimation

In the first regime, the neural network acts as a direct function approximator for a statistical estimator. The goal is to learn a mapping $f_\theta : \mathcal{X} \to \Theta$, parameterized by network weights θ, that outputs a point estimate $\widehat{\vartheta}$ given observed data x. Throughout this section, we use $\vartheta \in \Theta$ for the inferential target (a generic Bayesian parameter) and reserve θ for the trainable weights of the amortized estimator, matching the convention in Section 2. This formulation reduces inference to a supervised regression problem.

To ensure the network approximates a valid Bayesian estimator, we employ a decision-theoretic framework. We define a loss function $L(\vartheta, \widehat{\vartheta})$ that quantifies the cost of estimating the true parameter ϑ as $\widehat{\vartheta}$, and minimize the Bayes risk–the expected loss over the joint distribution of parameters and data. Let $\pi(\vartheta)$ denote the prior over parameters and $p(x \mid \vartheta)$ the likelihood. The Bayes risk $\mathcal{R}(\pi, f_\theta)$ is

$$\mathcal{R}(\pi, f_\theta) = \mathbb{E}_{\vartheta \sim \pi(\vartheta)} \left[\mathbb{E}_{x \sim p(x|\vartheta)} \left[L(\vartheta, f_\theta(x)) \right] \right], \tag{5}$$

or equivalently

$$\mathcal{R}(\pi, f_\theta) = \int_\Theta \int_\mathcal{X} L(\vartheta, f_\theta(x))\, p(x \mid \vartheta)\, \pi(\vartheta)\, dx\, d\vartheta.$$

Minimizing this risk with respect to θ yields the Bayes estimator under the chosen loss L. For instance, the squared-error loss $L(\vartheta, \widehat{\vartheta}) = \|\vartheta - \widehat{\vartheta}\|^2$ yields $f_{\theta^*}(x) \approx \mathbb{E}[\vartheta \mid x]$, the posterior mean.

3.1.1. Implicit topology learning as meta-learning

The capability of a neural network to perform statistical inference is not merely a curve-fitting exercise; it represents a form of meta-learning (Hospedales *et al.*, 2022). In traditional supervised learning, a model maps a specific input x to a corresponding label y. In the amortized inference regime, the network learns a functional mapping from an entire dataset $\mathcal{D}_n = \{x_1, \ldots, x_n\}$ to a parameter space Θ.

This learning mechanism requires the network to internalize the topology of the parameter manifold. Rather than memorizing answers for specific datasets, the network must learn the inductive biases of the generative model–it learns *to estimate*, not to recall. For example, when estimating the variance of a distribution, the network must implicitly learn that this parameter is strictly positive and that it correlates with the input dispersion, regardless of the specific input values.

Empirically, the latent representations learned by the network form a low-dimensional manifold aligned with the true parameter space Θ. This geometric alignment allows the

network to generalize to unseen datasets generated from the same prior, effectively replacing the manually derived formulas of classical estimation (such as the ordinary least squares formula) with a learned nonlinear equivalent.

3.2. Uncertainty quantification

While point estimation provides a single optimal summary of the parameter space–typically corresponding to Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP) approximations (Bishop, 2006; Goodfellow *et al.*, 2016), it is inherently insufficient for capturing the full predictive or structural distribution. In complex modeling tasks, a single point estimate fails to capture the epistemic uncertainty arising from finite training data or the aleatoric uncertainty inherent in the data-generating process. Consequently, moving beyond deterministic predictions requires a coherent framework for full uncertainty quantification (UQ).

Bayesian neural networks: Among the foundational paradigms for incorporating probabilistic reasoning into deep architectures is the Bayesian Neural Network (BNN) framework (MacKay, 1992; Neal, 1996). Instead of learning a single deterministic point estimate for the connection weights, a BNN assigns a prior probability distribution over the network parameters, denoted as $p(\mathbf{w})$. Given a training dataset $\mathcal{D}$, the inferential goal shifts to computing the posterior distribution over the weights, $p(\mathbf{w} \mid \mathcal{D})$, via Bayes' theorem. The predictive distribution for a novel test input $\mathbf{x}^*$ is subsequently evaluated by marginalizing over this parameter posterior:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \int p(y^* \mid \mathbf{x}^*, \mathbf{w}) p(\mathbf{w} \mid \mathcal{D})\, d\mathbf{w}$$

Despite their theoretical appeal, BNNs are computationally demanding and practically fragile. The true posterior over weights is analytically intractable because deep-network loss surfaces are high-dimensional and non-convex, so practitioners rely on structural or stochastic approximations. Sampling methods such as MCMC mix slowly in deep models and do not scale to architectures beyond shallow ones (Neal, 1996). Optimization-based approaches such as Variational Inference (VI; Graves, 2011; Blundell *et al.*, 2015) typically double the parameter count to model mean-field variances, introduce variance into gradient updates, and remain sensitive to hyperparameter tuning and initialization.

Deep ensembles: Another widely adopted paradigm for uncertainty quantification is the deep ensemble framework (Lakshminarayanan *et al.*, 2017). This approach involves training a collection of M independent neural networks, with each model initialized from a distinct random seed and exposed to different data shuffles or splits. At deployment, the epistemic uncertainty is quantified by evaluating the variance across the predictions or estimated quantities of the individual ensemble members. From a theoretical standpoint, because the optimization landscape of a deep neural network is highly non-convex and high-dimensional, varying the initialization trajectory forces each network to settle into distinct local minima. Consequently, the distributed outputs generated by these independent architectures can be viewed as empirical samples drawn from an implicit, multimodal posterior distribution, effectively capturing the geometric landscape of the model's predictive uncertainty.

The principal bottleneck is optimization cost: training time and memory scale linearly with the ensemble size M, so deep ensembles are difficult to scale in large networks or resource-constrained deployment environments where repeatedly retraining the model is infeasible.

Amortized variational inference: A third major paradigm for capturing uncertainty focuses on explicitly approximating the posterior distribution by restricting it to a tractable parametric family. Rather than executing independent optimization or sampling routines for each individual dataset, amortized variational inference (AVI) trains a global neural network to learn a direct functional mapping from the observed data to the parameters of a chosen variational distribution, $q_\phi(\boldsymbol{\theta} \mid \mathbf{x})$ (Kingma and Welling, 2014; Gershman and Goodman, 2014). During the upfront training phase, the network is optimized by maximizing the Evidence Lower Bound (ELBO) or minimizing an explicit probabilistic divergence (such as the Kullback-Leibler divergence) across a wide distribution of tasks. While traditional implementations often rely on simple mean-field Gaussian approximations, modern formulations extend this framework by integrating highly flexible density estimators, such as normalizing flows, to represent complex, skewed, or multimodal posterior surfaces (Rezende and Mohamed, 2015; Papamakarios *et al.*, 2021).

A primary advantage of this methodology is its computational efficiency during deployment. It shifts the intensive inductive workload into a single offline training phase, spreading computational costs across all subsequent inference tasks. After optimizing the global network, evaluating the full approximate posterior or generating samples for a new dataset requires only one rapid forward pass through the network. This approach avoids the slow, iterative updates of classical MCMC methods and eliminates the large storage requirements of ensemble techniques.

Uncertainty methodologies utilized in this study: To systematically evaluate and quantify uncertainty within our experimental analysis without incurring the computational overhead of deep ensembles or the optimization volatility of standard BNNs, we utilize three distinct methodologies: Monte Carlo Dropout, Monte Carlo Stability Analysis, and Flow-Based Posterior Sampling.

Monte carlo dropout: Dropout (Srivastava *et al.*, 2014) was introduced as a regularization technique used strictly during training: a random fraction of each layer's activations is zeroed at every step, which discourages co-adaptation of units and reduces overfitting. Gal and Ghahramani (2016) showed that retaining dropout at inference time yields a principled approximation to Bayesian inference. By keeping dropout active and repeating the forward pass on the same input, one obtains an empirical predictive distribution; the resulting stochastic forward passes are equivalent to variational inference in a Deep Gaussian Process, grounding a heuristic regularizer in formal inferential theory.

Formally, let $\widehat{\mathbf{W}}_l$ denote the trained weight matrix for layer $l \in \{1, \ldots, L\}$. At inference time, for each stochastic forward pass $t \in \{1, \ldots, T\}$, we sample a random binary mask vector from a Bernoulli distribution:

$$\mathbf{z}_{l,t} \sim \text{Bernoulli}(1 - q)$$

where q is the user-defined dropout probability. The active weights for the t-th realization are computed via the element-wise multiplication $\mathbf{W}_{l,t} = \widehat{\mathbf{W}}_l \cdot \mathrm{diag}(\mathbf{z}_{l,t})$. Given a novel test instance $\mathbf{x}^*$, the predictive posterior distribution is empirically approximated by averaging across the collection of T stochastic model trajectories:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} p(y^* \mid \mathbf{x}^*, \mathbf{W}_t)$$

From this sampled distribution, the epistemic uncertainty can be quantified by calculating the empirical variance across the T predictions.

Despite its convenience, Monte Carlo (MC) dropout has well-known limitations. As a variational approximation, it inherits the standard failure modes of mean-field VI: a tendency to underestimate posterior variance and an inability to represent multimodal weight distributions, both of which are constrained by the rigid Bernoulli masking. MC dropout is also sensitive to model capacity: in small or shallow architectures, removing a fixed fraction of units degrades predictive capacity rather than calibrating uncertainty.

Monte carlo stability analysis: To evaluate the consistency of our neural estimator, we employ a simulation-based stability analysis. We leverage the generative nature of our experimental framework to assess estimator variance across repeated realizations of the data-generation process.

For a fixed ground-truth p-length parameter vector β_{true} and a specific sample size N, we generate B independent datasets $\mathcal{D}_1, \ldots, \mathcal{D}_B$, where each $\mathcal{D}_b = \{(X^{(b)}, y^{(b)})\}$ for $b = 1, \ldots, B$. These datasets represent potential realizations of the world under identical underlying laws but different noise instances. We pass each dataset through the trained Transformer model to obtain a set of parameter estimates $\{\hat{\beta}^{(1)}, \ldots, \hat{\beta}^{(B)}\}$. It is essential to note that our model employs a hard-thresholding mechanism to enforce sparsity; the network produces both a magnitude vector and an inclusion probability. The final estimate is computed as the element-wise product of the magnitude and a binary mask derived from the probabilities (thresholded at 0.5). We quantify uncertainty by computing the standard deviation of these estimates across the B replications. The empirical stability metric is defined as the average standard deviation across all p coefficients:

$$\sigma_{\text{MC}}(N) = \frac{1}{p} \sum_{j=1}^{p} \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\beta}_j^{(b)} - \bar{\beta}_j\right)^2}. \tag{6}$$

By evaluating the metric in (6) across varying sample sizes (*e.g.*, $N \in \{50, \ldots, 1000\}$), we can empirically check the statistical consistency of the Transformer. A monotone decrease of σ_{MC} with N is consistent with a $\sqrt{N}$-style information-aggregation rate.

Flow-based posterior sampling: While the Monte Carlo stability analysis quantifies the stability of the point estimate, it does not capture the whole geometry of the posterior distribution, particularly in multimodal settings. To address this, we employ a Flow Matching framework to train a neural network as a generative model that draws samples directly from the posterior $p(\theta \mid x)$. Further, to overcome the computational bottlenecks of traditional

sampling, we leverage Conditional Flow Matching (CFM, Lipman *et al.*, 2023). In contrast to MCMC samplers, which rely on local iterative updates to explore the parameter space, our flow-based mechanism learns a deterministic transformation from a simple base distribution to the complex target distribution.

The core premise is to identify a transformation capable of mapping samples from a tractable initial distribution p_0 (*e.g.*, a standard Gaussian) to the intractable target posterior p_1. Flow matching models this transformation by identifying an ideal vector field that dictates the optimal movement of probability mass from the initial state to the final state over a time horizon $t \in [0, 1]$. We define a time-dependent vector field $v_t : \mathbb{R}^d \to \mathbb{R}^d$ parameterized by a neural network with parameters ϕ. The evolution of a sample z_t is governed by the ordinary differential equation

$$\frac{dz_t}{dt} = v_t(z_t; \phi).$$

By integrating this derivative from $t = 0$ to $t = 1$, the neural network effectively transports samples from the prior directly to the posterior.

During training, we regress the neural vector field v_t against a conditional target vector field u_t. We select the Optimal Transport path (Peyré and Cuturi, 2019), which creates a straight-line trajectory between a noise sample z_0 and a data sample z_1. The loss function minimizes the discrepancy between the learned field and this ideal trajectory

$$\mathcal{L}_{\text{CFM}}(\phi) = \mathbb{E}_{t,z_0,z_1}\left[\|v_t(z_t) - (z_1 - z_0)\|^2\right]. \tag{7}$$

This approach effectively amortizes the inference cost: once the ideal vector field is learned, sampling becomes a fast, non-iterative forward pass through the ODE solver, avoiding the slow mixing and mode collapse issues often encountered with MCMC in complex topologies.

4. Experimental evaluation

4.1. Experiment I: Latent structure recovery

We study task heterogeneity through a clustered task prior and evaluate whether permutation-invariant neural estimators can amortize inference by learning shared latent structure across tasks. Concretely, we construct a meta-learning environment where each task corresponds to a linear regression problem with a task-specific coefficient vector $\beta_t \in \mathbb{R}^p$, but the collection $\{\beta_t\}$ is constrained to lie on a finite set of latent centroids. We compare two amortized estimators, Deep Sets and Set Transformer, against a classical per-task estimator.

Generative process. We fix the feature dimension $p = 20$ and the number of latent clusters K. We first sample K latent centroids following $\mu_k \sim \mathcal{N}(0, \tau^2 I_p), k = 1, \ldots, K$ with $\tau = 3.0$. These centroids remain fixed for the entire experiment and define a latent task mixture. For each task t, we sample its regression coefficient vector by selecting one centroid uniformly at random, *i.e.*, $\beta_t \sim \text{Uniform}(\{\mu_1, \ldots, \mu_K\})$. Conditioned on β_t, we generate a task support set $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ with random support size

$N_t \sim \text{Discrete} - \text{Uniform}\{10, 11, \ldots, 30\}$. We sample the inputs as $x_{t,n} \overset{\text{IID}}{\sim} \mathcal{N}(0, I_p)$ and outputs follow the linear model

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n}, \qquad \varepsilon_{t,n} \overset{\text{IID}}{\sim} \mathcal{N}(0, \sigma^2), \qquad \sigma = 1.$$

Training and test meta-datasets. For each choice of $K \in \{5, 10, 50\}$, we independently generate a training meta-dataset of T_{train} tasks and a test meta-dataset of $T_{\text{test}} = 200$ held-out tasks using the same generative process (*i.e.*, the same K and hyperparameters). The neural models are trained across tasks to learn an amortized mapping

$$f_\theta : \ \mathcal{D}_t \mapsto \widehat{\beta}_t \in \mathbb{R}^p,$$

where f_θ is expressed as either a Deep Sets encoder or a Set Transformer encoder followed by a regression head. Both architectures operate on the set of paired observations $\{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ and are therefore permutation-invariant in the ordering of samples within a task.

Baselines and evaluation metric. As classical baselines, we compute a per-task Ordinary Least Squares (OLS) estimate $\widehat{\beta}_t^{\text{OLS}}$ and a Ridge regression estimate $\widehat{\beta}_t^{\text{Ridge}}$ using only the support set $\mathcal{D}_t$. We evaluate all estimators on the test tasks using parameter estimation error

$$\text{MSE}_\beta \ = \ \frac{1}{T_{\text{test}}} \sum_{t=1}^{T_{\text{test}}} \left\| \widehat{\beta}_t - \beta_t \right\|_2^2, \tag{8}$$

and train neural models for 100 epochs by minimizing the same objective over training tasks.

Parameter estimation error. Table 1 reports parameter-estimation error in the latent-structure-recovery experiment. The amortized neural estimators achieve parameter MSE one to two orders of magnitude below the per-task Ridge baseline. Linear baselines (OLS and Ridge) cannot exploit the shared geometry across tasks, so they suffer high error regardless of the cluster count K. Ridge improves marginally over OLS through ℓ_2 regularization, but remains fundamentally per-task. In low-complexity regimes ($K = 5, 10$), the neural estimators closely approximate the Bayes-optimal estimator; at $K = 50$, both architectures continue to generalize, indicating that they exploit the finite cluster structure of the parameter space even as it becomes more diffuse.

4.2. Experiment II: Distributional robustness

We next study the behavior of amortized estimators as the meta-training-set size grows, and their robustness to distributional mismatch. This experiment jointly evaluates (i) sample complexity in the number of training tasks and (ii) stability under increasingly non-Gaussian noise distributions. Unlike Experiment I, where task heterogeneity is induced through a clustered prior, here tasks are independently generated but exhibit substantial within-task uncertainty due to high-variance parameters and complex residual structure.

Table 1: Parameter estimation error (MSE_β in (8)), standard error (SE), and 95% coverage (Cov95) in the latent structure recovery experiment. Results are averaged over 200 held-out test tasks. For Ridge, we consider per-task linear estimators trained independently on each task using only its support set. For Deep Sets and Set Transformer, we consider amortized estimators trained across tasks.

	Ridge (per-task)	Deep Sets		Set Transformer	
Latent complexity	MSE	MSE ± SE	Cov95	MSE ± SE	Cov95
Clusters ($K = 5$)	5.027	0.248 ± 0.016	0.980	$\mathbf{0.043} \pm 0.002$	0.999
Clusters ($K = 10$)	5.245	1.031 ± 0.047	0.928	$\mathbf{0.102} \pm 0.007$	0.980
Clusters ($K = 50$)	4.822	6.242 ± 0.169	0.315	$\mathbf{2.370} \pm 0.100$	0.623

Generative process. We fix the feature dimension $p = 20$. For each task t, we draw regression coefficients independently from a high-variance prior $\beta_t \overset{\text{IID}}{\sim} \mathcal{N}(0, 9^2 I_p)$. Conditioned on β_t, we generate a support set $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ with inputs sampled as $x_{t,n} \overset{\text{IID}}{\sim} \mathcal{N}(0, I_p)$. We generate outputs according to the linear model

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n},$$

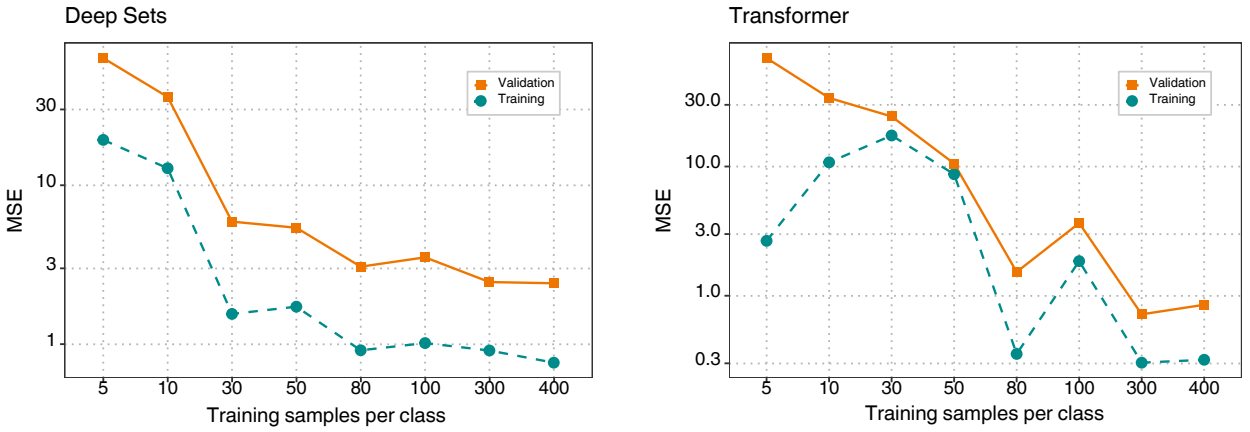
where the noise distribution $\varepsilon_{t,n}$ varies across regimes described below. To rigorously evaluate robustness to distributional mismatch, the final evaluation is performed on a held-out test set of $T_{\text{test}} = 600$ independent tasks for each noise regime.

Noise regimes. To assess robustness to distributional mismatch, we consider four residual distributions: (I) Gaussian (symmetric): $\varepsilon_{t,n} \sim \mathcal{N}(0, 1)$, (II) Asymmetric: $\varepsilon_{t,n} \overset{d}{=} E - 1$, where $E \sim \text{Exp}(1)$, (III) Bimodal: $\varepsilon_{t,n}$ follows a two-component Gaussian mixture with weights $\{0.8, 0.2\}$, and (IV) Trimodal: $\varepsilon_{t,n}$ follows a three-component Gaussian mixture with weights $\{0.8, 0.1, 0.1\}$. For the multimodal cases, component-specific means and variances are chosen to ensure well-separated modes, inducing heavy-tailed and non-convex error distributions.

Training and evaluation protocol. We generate a meta-dataset of up to $T = 6000$ tasks using the generative process above. From this pool, we form meta-training sets of size $T_{\text{train}} \in \{5, 10, 30, 50, 80, 100, 300, 400\}$ and a held-out test set of $T_{\text{test}} = 600$ tasks. For each value of T_{train}, we train Deep Sets and the Set Transformer to convergence using the same fixed optimization budget per fit, and evaluate the resulting estimator on the held-out test tasks.

This protocol examines sample complexity as a function of the meta-training-set size. Robustness to distributional mismatch is assessed by evaluating each trained model under every residual regime (Gaussian, Asymmetric, Bimodal, and Trimodal) while keeping the task prior and the task sample-size distribution fixed.

Objective and metrics. As in Experiment I, Deep Sets and Set Transformer architectures learn an amortized mapping $f_\theta : \mathcal{D}_t \mapsto \widehat{\beta}_t \in \mathbb{R}^p$. We evaluate the performance using



(a) Deep Sets **(b) Set Transformer**

Figure 4: Sample complexity in the meta-training-set size: dashed lines report training-task fit and solid lines report held-out test error, as a function of the number of training tasks T_{train}. (a) Deep Sets achieves low error even with small $T_{\text{train}} < 50$ but plateaus quickly. (b) The Set Transformer requires more training tasks ($T_{\text{train}} > 80$) to stabilize, but reaches a substantially lower asymptotic error.

the parameter estimation error $\text{MSE}_\beta = \mathbb{E}\left[\|\widehat{\beta}_t - \beta_t\|_2^2\right]$, computed over held-out tasks and averaged across noise regimes and support sizes.

Sample complexity in the meta-training-set size. Figure 4 reports the training-task fit and held-out test error of Deep Sets and the Set Transformer as the meta-training-set size T_{train} is varied over $\{5, 10, 30, 50, 80, 100, 300, 400\}$. The horizontal axis is the number of training tasks on which the amortized estimator was fit. It is distinct from the optimization budget (held fixed per fit) and from the per-task sample size N_t (drawn from a fixed distribution).

Deep Sets (left panel) attains low test error with only a small number of training tasks; both training and held-out error decrease sharply for $T_{\text{train}} \leq 30$–50 and improve only marginally thereafter. The Set Transformer (right panel) has higher error and greater variance in the small-T_{train} regime, but its error continues to decrease as more training tasks are supplied. By $T_{\text{train}} \geq 300$, the Set Transformer reaches a substantially lower asymptotic error than Deep Sets. In summary, Deep Sets is sample-efficient in T_{train} but capacity-limited; the Set Transformer requires more training tasks to identify but attains a better solution.

Complex noise robustness. To systematically evaluate the robustness of the models, we injected different noise profiles into the dataset. The noise distributions in Table 2 are categorized into three distinct groups: *Clean* (symmetric, baseline Gaussian noise), *Skewed* (asymmetric, heavy-tailed noise that violates standard normality assumptions), and *Multimodal* (mixture models introducing distinct, unequally weighted error clusters).

Training the Set Transformer is more demanding than training simpler architectures: it requires a larger meta-training set and careful hyperparameter tuning, and we observed slower and less stable convergence under heavy-tailed and multimodal noise. The high coverage reported for the Set Transformer in Table 2 is partly explained by the wider predictive

Table 2: Parameter estimation error ($\text{MSE} \pm \text{SE}$) **and 95% coverage across clean, skewed, and multimodal noise distributions. *Clean Small* and *Clean High* denote Gaussian residuals with low and high variance ($\sigma = 1$ and $\sigma = 3$). *Right/Left Skew (r1)* and *(r2)* are shifted Exponential residuals with rate parameters $r_1 = 1$ and $r_2 = 0.5$ (heavier-tailed). *Bi-Unbalanced ($m\mu s\sigma$)* denotes a two-component Gaussian mixture with weights $\{0.8, 0.2\}$ whose minority component has mean μ and standard deviation σ. *Trimodal Tight* and *Trimodal Wide* denote three-component Gaussian mixtures with weights $\{0.8, 0.1, 0.1\}$ whose modes are closely or widely separated, respectively.**

	Deep Sets		Set Transformer	
Noise Type	MSE ± SE	Cov95	MSE ± SE	Cov95
Clean Distributions				
Clean Small	6.86 ± 0.357	0.980	1.72 ± 0.100	0.984
Clean High	7.17 ± 0.341	0.974	1.56 ± 0.081	0.991
Skewed Distributions				
Right Skew (r1)	7.61 ± 0.393	0.972	1.65 ± 0.092	0.988
Left Skew (r1)	7.20 ± 0.340	0.975	1.63 ± 0.100	0.989
Right Skew (r2)	6.78 ± 0.339	0.977	1.54 ± 0.082	0.989
Left Skew (r2)	8.00 ± 0.404	0.974	1.73 ± 0.108	0.986
Multimodal Distributions				
Bi-Unbalanced (m3s1)	7.96 ± 0.414	0.972	2.21 ± 0.125	0.977
Bi-Unbalanced (m4s2)	9.99 ± 0.451	0.953	3.36 ± 0.201	0.955
Trimodal Tight	8.00 ± 0.377	0.975	1.95 ± 0.112	0.985
Trimodal Wide	7.42 ± 0.385	0.970	1.72 ± 0.089	0.990

variance due to this weight sensitivity, as reflected in the larger standard errors of its MSE estimates.

As shown in Figure 5, we assess the stability of the amortized estimators by measuring Monte Carlo variability as a function of the support set size N. Here, N denotes the number of observations provided to the model for a single task at inference time, and should be clearly distinguished from the meta-training-set size T_{train} or the optimization budget. For each task and noise regime, we draw replicated samples over the support set and compute the standard deviation of the resulting parameter estimates. This quantity captures the sensitivity of the estimator to sampling variability and provides a direct measure of predictive stability. Both architectures exhibit a clear monotonic decrease in Monte Carlo standard deviation as N increases, indicating that parameter estimates become more stable as additional evidence is provided. This behavior is consistent with that of well-behaved statistical estimators, where the uncertainty decreases as the information content of the observed data increases. Notably, the Set Transformer demonstrates substantially lower uncertainty across all noise regimes and support sizes. In particular, its Monte Carlo standard deviation remains below 0.6 even in low-data settings, whereas Deep Sets exhibits considerably higher variability, with deviations exceeding 1.5 for small N. Moreover, the Transformer's uncertainty curves for Gaussian, asymmetric, and multimodal noise remain closely aligned, suggesting that its stability is largely insensitive to the form of the noise distribution. This observation indicates that

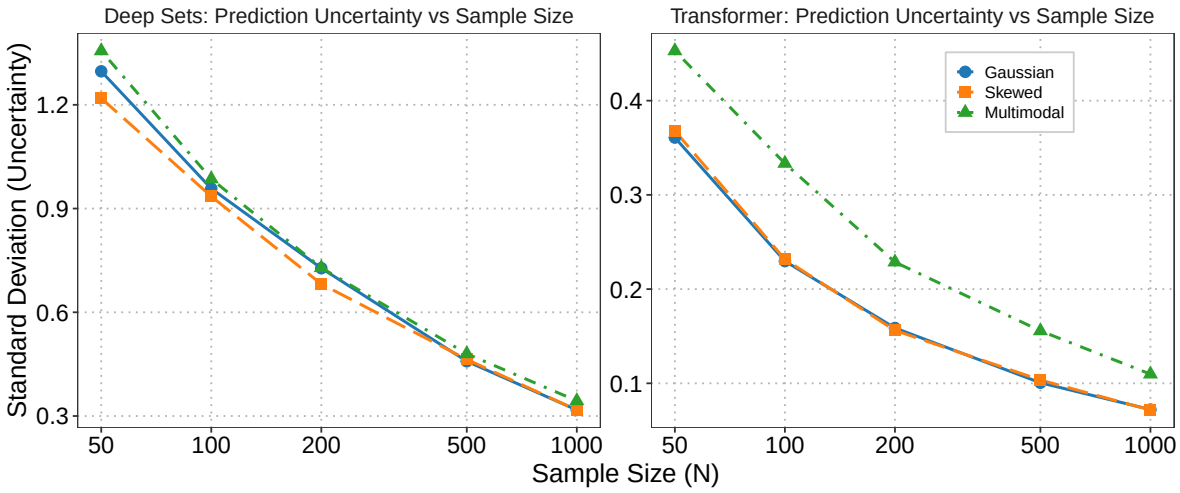


Figure 5: Monte Carlo stability analysis. Standard deviation of the estimated regression coefficients (computed via replicated sampling of the task support set) as a function of support set size N. Both architectures exhibit decreasing uncertainty as N increases.

the attention-based architecture produces more reliable and consistent parameter estimates under both data scarcity and distributional mismatch.

4.3. Experiment III: Sparse signal recovery

To further probe Regime II (the data-scarce frontier), we study the recovery of sparse signals in high-dimensional linear regression. Each task corresponds to estimating a task-specific sparse coefficient vector from a small set of observations. Our goal is to learn an amortized, permutation-invariant estimator that identifies both the support and magnitudes of the sparse signal.

Generative process. We fix the feature dimension $p = 100$. For each task t, we generate a sparse regression coefficient vector $\beta_t \in \mathbb{R}^p$ with a sparsity level k_t denoting the *percentage of zero elements*, so that

$$\|\beta_t\|_0 = p - k_t, \qquad k_t \in \mathcal{K} := \{5, 10, 15, \ldots, 100\}.$$

The active support $S_t \subset \{1, \ldots, p\}$ with $|S_t| = p - k_t$ is sampled uniformly at random (*e.g.*, $k_t = 95$ leaves only 5 active variables). Conditional on S_t, the nonzero coefficients are drawn independently as $(\beta_t)_j \sim \mathcal{N}(0, 3), \quad j \in S_t$, with $(\beta_t)_j = 0$ for $j \notin S_t$. Conditioned on β_t, we generate a task-specific support set $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$, where covariates are sampled as $x_{t,n} \sim \mathcal{N}(0, I_p)$ and responses follow the linear model

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n}, \qquad \varepsilon_{t,n} \sim \mathcal{N}(0, 1).$$

The number of observations per task is drawn uniformly as

$$N_t \sim \text{Discrete} - \text{Uniform}\{400, 401, \ldots, 500\}.$$

Model. For each task t, we treat $\mathcal{D}_t$ as an unordered set of tokens $z_{t,n} = [x_{t,n}; y_{t,n}] \in \mathbb{R}^{p+1}$. Tokens are embedded into $\mathbb{R}^{d_{\text{model}}}$ and processed by an L-layer Transformer encoder 'without positional encodings', ensuring permutation equivariance across samples. A mean pooling operator produces a permutation-invariant task representation r_t, which is decoded by two heads: a magnitude head $\widehat{\beta}_{\text{mag},t} \in \mathbb{R}^p$ and a sparsity gate $\hat{p}_t \in (0,1)^p$. The final coefficient estimate is

$$\widehat{\beta}_t = \widehat{\beta}_{\text{mag},t} \odot \hat{p}_t.$$

Training objective. Given a batch of M tasks, we minimize prediction error using soft gating:

$$\mathcal{L} = \frac{1}{M} \sum_{t=1}^{M} \left(\frac{1}{N_t} \sum_{n=1}^{N_t} \left(y_{t,n} - x_{t,n}^\top \widehat{\beta}_t \right)^2 \right).$$

Training meta-dataset construction. We generate a fixed meta-dataset of $T = 6000$ tasks spanning the grid of sparsity levels $\mathcal{K} = \{5, 10, 15, \ldots, 100\}$ ($|\mathcal{K}| = 20$). For each $k \in \mathcal{K}$, we generate 300 independent tasks, yielding $T = 20 \times 300 = 6000$ tasks in total. Each task t is assigned a sparsity level $k_t \in \mathcal{K}$ and is generated according to the sparse linear model described above.

Per-task sampling. For a task with sparsity level k_t (percentage of zero coefficients), we sample an active support $S_t \subset \{1, \ldots, p\}$ with $|S_t| = p - k_t$ uniformly at random, draw nonzero coefficients $(\beta_t)_j \sim \mathcal{N}(0,1)$ for $j \in S_t$ (and $(\beta_t)_j = 0$ otherwise), and then generate a task-specific dataset $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$. The number of observations per task is drawn uniformly as $N_t \sim \text{Discrete} - \text{Uniform}\{400, 401, \ldots, 500\}$, with covariates sampled as $x_{t,n} \overset{\text{IID}}{\sim} \mathcal{N}(0, I_p)$ and responses generated as

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n}, \qquad \varepsilon_{t,n} \sim \mathcal{N}(0,1).$$

Train–test split. After generating the full collection of $T = 6000$ tasks, we randomly shuffle task indices and split tasks into $T_{\text{train}} = 5400$ training tasks and $T_{\text{test}} = 600$ held-out test tasks. All models are trained only on the training tasks; evaluation is performed on the held-out tasks.

Performance. Figure 6 reports the across-epoch performance of the sparse recovery architecture under varying sparsity levels compared against the Lasso baseline. We measure cosine similarity between the predicted coefficient vector $\widehat{\boldsymbol{\beta}}$ and the ground-truth $\boldsymbol{\beta}$, where the sparsity parameter k denotes the percentage of zero coefficients in $\boldsymbol{\beta}$. The Lasso baseline demonstrates exceptional point-estimation accuracy, consistently achieving a cosine similarity close to 1 across all sparsity regimes. For our neural model, moderate sparsity levels (*e.g.*, $k \leq 60\%$) yield high cosine similarity (> 0.85) even at early training stages (epoch 100), indicating accurate recovery of the dominant signal components. As sparsity increases (larger k), the recovery task becomes more challenging due to fewer informative features, leading to lower initial performance at epoch 100. However, continued optimization leads to consistent improvements, and by epoch 500, the model closely approaches the baseline

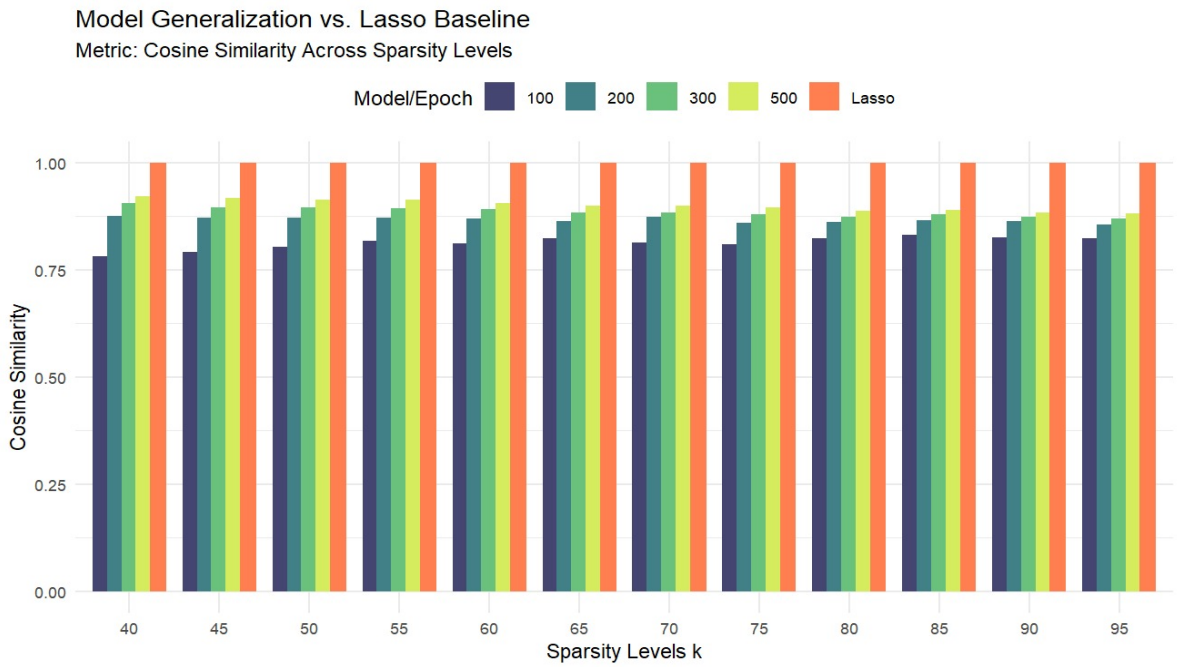


Figure 6: Cosine similarity between estimated and true regression coefficients across different sparsity levels k, where $k\%$ of the coefficients in the ground-truth parameter vector are zero. Bar clusters compare the neural network's optimization progression at training epochs 100, 200, 300, and 500 against the baseline Lasso regression performance.

performance. Crucially, while Lasso solves an independent optimization problem for each task at inference time, the amortized model produces estimates in a single forward pass, making it well-suited to large-scale multi-task deployment.

While cosine similarity captures point-estimation accuracy, it does not reflect the stability of the recovered coefficients across different data realizations. To assess the reliability of the estimator beyond point estimates, we analyze the empirical standard deviation of the predicted coefficients, denoted as $\sigma(\widehat{\boldsymbol{\beta}})$. Table 3 reports this quantity as a function of the support set size N across sparsity percentages $k \in \{20, 50, 80, 95\}$. In low-sample regimes ($N = 50$), uncertainty is elevated across all sparsity levels, particularly when sparsity is high, reflecting the difficulty of disentangling signal from noise with limited observations. As the number of observations increases, the standard deviation decreases monotonically for all sparsity levels. At $N = 1000$, uncertainty is uniformly low, indicating stable coefficient recovery even under extreme sparsity.

4.4. Experiment IV: Topological mismatch in multimodality

We next consider a setting in which the posterior distribution over task parameters exhibits a substantial topological mismatch relative to standard unimodal assumptions. In particular, we study amortized inference when the parameter distribution is highly multimodal and supported on a non-convex manifold. This experiment evaluates whether neural inference models can identify and represent disconnected posterior geometry using only observed input–output samples.

Table 3: Standard deviation of predicted regression coefficients $\sigma(\widehat{\beta})$ across varying support set sizes (N) and sparsity percentages (k). Here, k denotes the percentage of zero coefficients in the true parameter vector. The values reported here are calculated based on averaging the standard deviations obtained from 30 replications.

Support set size (N)	Sparsity percentage (k)			
	20	50	80	95
50	0.530	0.498	0.445	0.412
100	0.392	0.355	0.310	0.285
200	0.258	0.220	0.185	0.150
500	0.140	0.125	0.095	0.085
1000	0.062	0.055	0.048	0.042

Generative model. Let $\boldsymbol{\beta} \in \mathbb{R}^2$ denote the task-specific regression parameter. We define a multimodal prior distribution consisting of $K = 8$ Gaussian components arranged uniformly on a circle of radius $R = 5.0$:

$$p(\boldsymbol{\beta}) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\boldsymbol{\beta} \mid \boldsymbol{\mu}_k, \sigma^2 I_2), \qquad \boldsymbol{\mu}_k = R\begin{bmatrix}\cos\left(\frac{2\pi k}{K}\right)\\ \sin\left(\frac{2\pi k}{K}\right)\end{bmatrix}.$$

Conditioned on $\boldsymbol{\beta}$, observations are generated according to the same linear model used throughout the paper

$$y_n = x_n^\top \boldsymbol{\beta} + \varepsilon_n, \qquad x_n \sim \mathcal{N}(0, I_2),\ \varepsilon_n \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

yielding a task-specific dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$.

Inference objective. Given $\mathcal{D}$, the goal is to approximate the posterior distribution $p(\boldsymbol{\beta} \mid \mathcal{D})$, which inherits the multimodal structure of the prior while being shaped by the likelihood induced by the observed (x, y) pairs. Notably, the inference model has access only to the observed samples and must identify the full posterior geometry without explicit knowledge of the prior components.

Amortized conditional flow. We employ a conditional normalizing flow trained via Conditional Flow Matching to learn a mapping from observed datasets to posterior distributions. The input dataset $\mathcal{D}$ is first embedded into a fixed-dimensional context vector

$$\boldsymbol{r} = f_\psi(\mathcal{D}),$$

using a permutation-invariant encoder consistent with earlier experiments (Deep Sets). This representation summarizes the information contained in the observed (x, y) samples and conditions the inference model.

Let $\boldsymbol{\beta}_t \in \mathbb{R}^2$ denote the state of the flow at time $t \in [0, 1]$, initialized from a base distribution $\boldsymbol{\beta}_0 \sim \mathcal{N}(0, I_2)$. The evolution of $\boldsymbol{\beta}_t$ is governed by a time-dependent velocity field

$$\frac{d\boldsymbol{\beta}_t}{dt} = v_\phi(t, \boldsymbol{\beta}_t, \boldsymbol{r}), \tag{9}$$

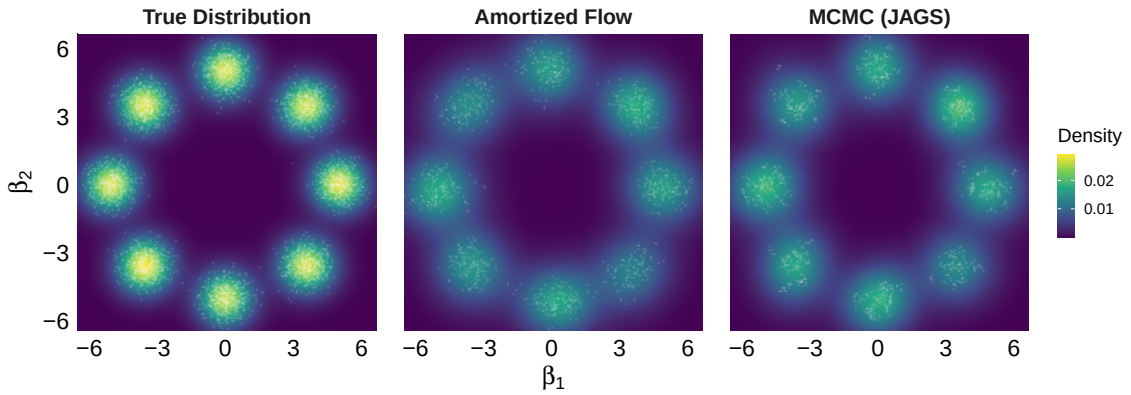


Figure 7: Posterior Sampling Benchmark: Visual comparison of the true multimodal distribution (Left), the Amortized Normalizing Flow approximation (Center), and the MCMC baseline (Right). The Amortized Flow successfully captures the complex multimodal geometry and separates the modes significantly faster than the baseline.

where v_ϕ is a neural network parameterized by ϕ. Integrating this ordinary differential equation yields

$$\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 + \int_0^1 v_\phi(t, \boldsymbol{\beta}_t, \boldsymbol{r})\, dt, \tag{10}$$

which represents a sample from the learned approximation to $p(\boldsymbol{\beta} \mid \mathcal{D})$.

Posterior geometry and transport dynamics. Figure 7 compares samples from the true posterior distribution, the amortized flow approximation, and an MCMC baseline. The amortized model successfully allocates probability mass across all eight modes, demonstrating that it recovers the disconnected geometry of the posterior using only the observed (x, y) samples. To quantitatively compare the posterior approximations, we compute the Kullback–Leibler (KL) and Jensen–Shannon (JS) divergences between the true posterior and each approximation. The amortized flow yields a KL divergence of 0.2728 and a JS divergence of 0.0587, while the corresponding values for the MCMC approximation are 0.2544 and 0.0528, respectively.

Figure 8 visualizes the transport induced by the learned vector field. Samples drawn from the isotropic Gaussian base distribution are smoothly transformed over time into the multimodal posterior structure, confirming that the model learns a continuous deformation that respects the topology of the target distribution.

Computational efficiency. In addition to capturing complex posterior geometry, the amortized flow provides substantial computational benefits. Posterior samples are generated in 0.82 seconds per task, compared to 2.76 seconds for the MCMC baseline. Although the MCMC approximation is marginally closer to the true posterior according to these divergence measures, the differences are relatively small. In contrast, the amortized flow provides posterior samples through a single forward pass after training, resulting in substantially lower computational cost than running an MCMC algorithm for each new dataset.

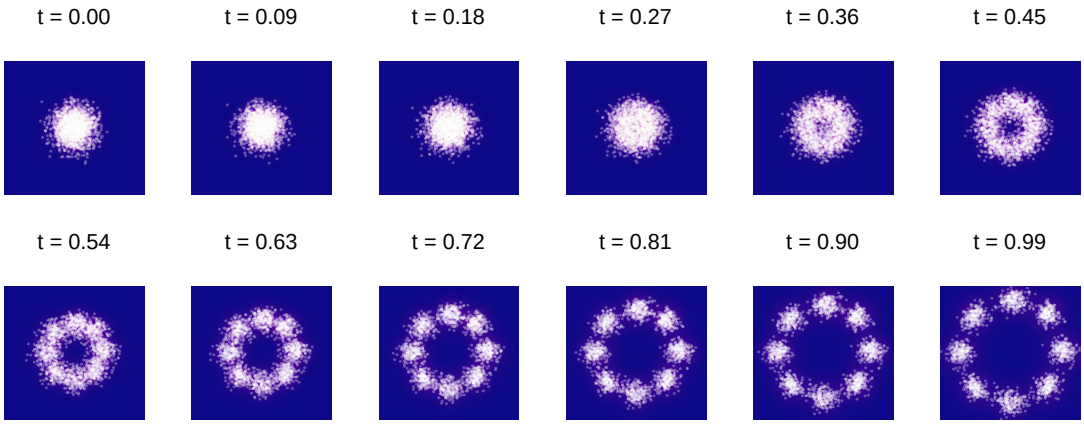


Figure 8: Visualization of the flow trajectory evolution. The horizontal panels illustrate the transformation of the initial particle distribution ($\theta_0 \sim \mathcal{N}(0, I)$ at $t = 0.00$) under the learned vector field over time. By $t = 1.00$, the particles have successfully converged to the target 8-mode posterior geometry.

5. Discussion and future research

Our experimental evaluation highlights the practical trade-offs of using permutation-invariant neural architectures for amortized inference. The primary advantage of these models is their computational efficiency during inference. As demonstrated in our sparse signal recovery (Section 4.3) and multimodal posterior experiments (Section 4.4), amortized models replace computationally expensive per-task optimizations, such as Lasso or MCMC, with a single forward pass. This allows for highly scalable inference, reducing posterior sampling time from 2.76 seconds to 0.82 seconds per task.

Furthermore, these architectures excel when tasks share a hidden structural geometry. In the latent structure recovery experiment (Section 4.1), both Deep Sets and the Set Transformer successfully pooled information across tasks to uncover hidden clusters, achieving parameter estimation errors substantially lower than those of classical per-task linear baselines (*e.g.*, Ridge regression) that fail to leverage cross-task information.

However, our results also emphasize that model selection must be carefully tailored to the specific application, as significant differences exist in training dynamics and robustness. Deep Sets provides rapid training convergence, reaching stable validation errors in fewer than 50 epochs (Section 4.2). In contrast, the Set Transformer requires extended optimization (over 80 epochs) to stabilize but ultimately achieves a lower asymptotic error. While the Transformer demonstrates superior Monte Carlo stability as the support set size increases, its architecture inherently requires larger data volumes and more intricate hyperparameter tuning. As observed in our distributional robustness evaluations, the Transformer is highly sensitive to its weights, leading to high prediction variance and difficulty generalizing across diverse, complex noise profiles.

Finally, these findings indicate that amortized neural inference is particularly promising for domains where classical distributional assumptions fail. The amortized flow model

successfully identified and represented a disconnected, non-convex, multimodal posterior without relying on unimodal approximations (Section 4.4). Future research will focus on scaling these architectures to higher-dimensional, real-world datasets characterized by extreme data sparsity and complex topological mismatches, investigating methods to stabilize Transformer training under heavy-tailed and asymmetric noise. Specifically, the use of MCMC is common in the literature focusing on large spatial and spatiotemporal data modeling using Gaussian processes, *e.g.*, under geostatistics (Gelfand and Schliep, 2016; Hazra *et al.*, 2025b), multi-type processes (Cisneros *et al.*, 2023; Mukherjee *et al.*, 2026), spatially varying coefficients models (Gelfand *et al.*, 2003; Bhowmik and Hazra, 2026), heavy-tailed processes (Hazra *et al.*, 2020, 2025a), nonstationary processes (Hazra and Huser, 2021), longitudinal datasets (Hazra *et al.*, 2019), and extreme value analysis (Jóhannesson *et al.*, 2022; Hazra *et al.*, 2023); amortized Bayesian inference is an emerging area of Bayesian statistics potentially suitable to handle such computationally challenging problems efficiently.

Acknowledgement

The authors would like to thank the Handling Editor and an anonymous Reviewer for their detailed constructive feedback, which helped to improve the quality and flow of the manuscript. The authors acknowledge the use of ChatGPT (OpenAI) and Grammarly for assistance with grammar correction, language refinement, and sentence restructuring. The scientific content, analysis, results, and conclusions presented in this paper are solely the responsibility of the authors.

Conflict of interest

The authors do not have any financial or non-financial conflicts of interest to declare for the research work included in this article.

References

Andoni, A. and Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, **51**, 117–122.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2018). Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, **39**, 930–945.

Bhowmik, S. and Hazra, A. (2026). A Bayesian latent Gaussian conditional autoregressive copula model for analyzing spatially-varying trends in rainfall. *Japanese Journal of Statistics and Data Science*, **9**, 99–131.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, **16**, 199–231.

Cisneros, D., Gong, Y., Yadav, R., Hazra, A., and Huser, R. (2023). A combined statistical and machine learning approach for spatial prediction of extreme wildfire frequencies and sizes. *Extremes*, **26**, 301–330.

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

Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian processes. In *Artificial Intelligence and Statistics (AISTATS)*, pages 207–215. PMLR.

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

Gelfand, A. E., Kim, H.-J., Sirmans, C., and Banerjee, S. (2003). Spatial modeling with spatially varying coefficient processes. *Journal of the American Statistical Association*, **98**, 387–396.

Gelfand, A. E. and Schliep, E. M. (2016). Spatial statistics and Gaussian processes: A beautiful marriage. *Spatial Statistics*, **18**, 86–104.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, **85**, 398–409.

Gershman, S. J. and Goodman, N. D. (2014). Amortized inference in probabilistic reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356.

Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR.

Hazra, A. and Huser, R. (2021). Estimating high-resolution Red Sea surface temperature hotspots, using a low-rank semiparametric spatial model. *The Annals of Applied Statistics*, **15**, 572–596.

Hazra, A., Huser, R., and Bolin, D. (2025a). Efficient modeling of spatial extremes over large geographical domains. *Journal of Computational and Graphical Statistics*, **34**, 795–811.

Hazra, A., Huser, R., and Jóhannesson, Á. V. (2023). Bayesian latent Gaussian models for high-dimensional spatial extremes. In *Statistical Modeling Using Bayesian Latent Gaussian Models: With Applications in Geophysics and Environmental Sciences*, pages 219–251. Springer.

Hazra, A., Nag, P., Yadav, R., and Sun, Y. (2025b). Exploring the efficacy of statistical and deep learning methods for large spatial datasets: A case study. *Journal of Agricultural, Biological and Environmental Statistics*, **30**, 231–254.

Hazra, A., Reich, B. J., Reich, D. S., Shinohara, R. T., and Staicu, A.-M. (2019). A spatio-temporal model for longitudinal image-on-image regression. *Statistics in Biosciences*, **11**, 22–46.

Hazra, A., Reich, B. J., and Staicu, A.-M. (2020). A multivariate spatial skew-t process for joint modeling of extreme precipitation indexes. *Environmetrics*, **31**, e2602.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). NeuTra-lizing Bad Geometry in Hamiltonian Monte Carlo Using Neural Transport. *arXiv preprint arXiv:1903.03704*, .

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, 251–257.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2022). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**, 5149–5169.

Jóhannesson, Á. V., Siegert, S., Huser, R., Bakka, H., and Hrafnkelsson, B. (2022). Approximate Bayesian inference for analysis of spatiotemporal flood frequency data. *The Annals of Applied Statistics*, **16**, 905–935.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Kitaev, N., Kaiser, &., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In *International Conference on Learning Representations*.

Luccioni, A. S., Viguier, S., and Ligozat, A.-L. (2023). Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *Journal of Machine Learning Research*, **24**, 1–15.

MacKay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, **4**, 448–472.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, volume 27.

Mukherjee, A., Hazra, A., and Vats, D. (2026). Scalable Bayesian inference for high-dimensional mixed-type multivariate spatial data. *Spatial Statistics*, **74**, 101020.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, **9**, 141–142.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, .

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, **22**, 1–64.

Peyré, G. and Cuturi, M. (2019). Computational optimal transport with applications to data sciences. *Foundations and Trends® in Machine Learning*, **11**, 355–607.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, .

Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köthe, U., and Bürkner, P.-C. (2024). Neural methods for amortised parameter inference. *arXiv:2404.12484*, .

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538.

Richards, J., Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2024). Neural Bayes estimators for censored inference with peaks-over-threshold models. *Journal of Machine Learning Research*, **25**, 1–49.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **71**, 319–392.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2024). Likelihood-free parameter estimation with neural Bayes estimators. *The American Statistician*, **78**, 1–14.

Sainsbury-Dale, M., Zammit-Mangion, A., Richards, J., and Huser, R. (2025). Neural Bayes estimators for irregular spatial data using graph neural networks. *Journal of Computational and Graphical Statistics*, **34**, 1153–1168.

Shmueli, G. (2010). To explain or to predict? *Statistical Science*, **25**, 289–310.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, **15**, 1929–1958.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Wagstaff, E., Fuchs, F., Engelcke, M., Posner, I., and Osborne, M. A. (2019). On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pages 6487–6494. PMLR.

Wand, M. P. and Jones, M. C. (1994). *Kernel Smoothing*. CRC press.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, **26**, 359–372.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30.

Zammit-Mangion, A., Sainsbury-Dale, M., and Huser, R. (2025). Neural methods for amortized inference. *Annual Review of Statistics and Its Application*, **12**, 311–335.