Bayesian Last Layer for Neural Force Fields

Anonymous Authors[1]

Abstract

Reliable uncertainty quantification is essential for deploying Machine Learning Interatomic Potentials (MLIPs), also known as Neural Force Fields, especially when molecular dynamics or materials simulations encounter configurations outside the training distribution. Deep ensembles remain the strongest practical baseline for MLIP uncertainty, but training and storing several copies of a modern pretrained model is often prohibitively expensive. We show that Bayesian Linear Last Layers (BLLs) provide a scalable alternative for MLIPs: a single pretrained backbone supplies atomic features, while exact Bayesian inference over the final force-prediction layer gives predictive uncertainties. BLL is known to underestimate the uncertainties. We provide an in-depth analysis that shows sources of miscalibration and introduce a simple post-hoc recalibration to address the issue. On MPtrj and rMD17 benchmarks, including both in-distribution tests and increasingly out-of-distribution regimes, BLLs that are recalibrated on in-distribution examples produce uncertainty estimates competitive with ensembles, while using only one base model.

Computational chemistry and materials science are currently undergoing a structural transformation driven by Machine Learning Interatomic Potentials (MLIPs), also known as Neural Force Fields. While ab initio methods like Density Functional Theory (DFT) provide high accuracy, their cubic scaling with system size makes them prohibitively expensive for large-scale systems or long-duration molecular dynamics (MD) simulations. MLIPs bridge this gap by learning to map atomic configurations directly onto the potential energy surface, offering DFT-level accuracy at a fraction of the cost. Recent architectural innovations and the emergence of massive, heterogeneous datasets have led to "universal" foundation models, such as the Orb-v3 (Rhodes et al., 2025) and eSEN (Fu et al., 2025) architecture, which further motivate the neural approach.

However, the neural surrogate approach is highly susceptible to catastrophic failures when encountering out-of-distribution (OOD) atomic configurations. This vulnerability is particularly acute in MD simulations; a single unphysical prediction can drive a system into high-energy distorted states that were not present in the training manifold and create a spiral where the model is thrown further into OOD regime.

Uncertainty Quantification (UQ) offers a path forward. Using UQ, one could identify inaccurate predictions before deployment, potentially triggering a more expensive, more accurate computation for a given configuration (Pilania et al., 2017). Moreover, Bayesian uncertainties provide a natural pathway within an active learning loop, in which accurate DFT calculations can be used to further improve the learning-based approach.

Unfortunately, UQ remains an effectively unresolved problem in deep learning (Papamarkou et al., 2024), where the practical gold standard, deep ensembles (Tan et al., 2023), are prohibitively costly at the scale of foundation models. This paper investigates the feasibility of Bayesian Last Layer (BLL) in MLIPs, in which exact Bayesian posterior estimation is limited to the final linear head.

While simple and elegant, BLL is known to underestimate the uncertainty. We provide an analysis that suggests a decoupled recalibration of aleatoric and epistemic terms in BLL. As a result, we report that BLL favourably competes with deep ensembles at a much lower cost, effectively positioning it as an effective tool for MLIPs.

1. Related Works

Uncertainty quantification in MLIPs. The need for reliable uncertainty estimates in MLIPs has been recognized since their early days (Bartók et al., 2010). The dominant approach remains deep ensembles (Lakshminarayanan et al., 2017), in which multiple independently trained models provide both a mean prediction and a measure of disagreement. A systematic comparison by Tan et al. (2023) across several atomic force and energy benchmarks found that ensembles consistently outperformed single-model alternatives

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Submitted to the AI for Science workshop (ICML 2026)

— including Gaussian mixture models, deep evidential regression, and mean-variance estimation — which tended to fail in specific regimes. This has cemented ensembles as the practical gold standard for UQ in MLIPs. Training and storing multiple copies of large foundation models such as UMA (Wood et al., 2025), Orb-v3 (Rhodes et al., 2025), and eSEN (Fu et al., 2025) is, however, prohibitive at scale, motivating the search for cheaper single-model alternatives that can match ensemble-level calibration.

Bayesian neural networks. BNNs provide a principled framework for UQ by maintaining a posterior over model parameters (MacKay, 1992; Neal, 1996), with approximate inference via MCMC (Izmailov et al., 2021), variational inference (Blundell et al., 2015; Graves, 2011; Thaler et al., 2023), and Laplace approximations (Ritter et al., 2018; Daxberger et al., 2021). Several works draw formal connections between deep ensembles and Bayesian inference (Wilson & Izmailov, 2020; Pearce et al., 2020), suggesting that the empirical success of ensembles partly reflects implicit Bayesian properties.

Gaussian processes for interatomic potentials. A natural compromise between full BNNs and point-estimate networks is to apply Bayesian inference to features extracted by the deep network. The Gaussian Approximation Potential (GAP) framework (Bartók et al., 2010) pairs Gaussian processes (GPs) with handcrafted atomic descriptors such as SOAP (Bartók et al., 2013) to learn potential energy surfaces with native uncertainty estimates. GPs are attractive in this setting because exact posterior inference is available in closed form, but their non-parametric nature scales as $O(N^3)$ in the dataset size and limits their applicability to the large, heterogeneous datasets underpinning modern foundation MLIPs.

Bayesian last-layer methods. BLL retains the closed-form posterior of GPs but shifts the cost from N to the feature dimension D, making it tractable at MLIP scale; it is mathematically equivalent to a GP with a linear kernel in the learned feature space (Rasmussen & Williams, 2006). Watson et al. (2021) showed that BLLs can outperform full BNNs trained with Bayes-by-Backprop on standard regression tasks. The principal limitation of last-layer methods is feature collapse, the tendency of deterministic feature extractors to map dissimilar inputs to nearby representations, which can render the last-layer posterior overconfident on out-of-distribution inputs (van Amersfoort et al., 2021). Existing remedies modify the backbone, e.g., spectral normalization in SNGP (Liu et al., 2020) and DUQ (van Amersfoort et al., 2020). We instead operate in the foundation-MLIP regime in which the backbone is taken as a fixed point estimate, address calibration post hoc, and characterize the residual error this leaves uncorrected (Section 3.2).

Post-hoc calibration. Post-hoc methods adjust pretrained model uncertainties using held-out data, with temperature scaling (Guo et al., 2017) the canonical approach in classification. Calibration in regression has received comparatively less attention (Kuleshov et al., 2018). The generalized posterior and SafeBayes literature (Grünwald & van Ommen, 2018; Bissiri et al., 2016) provides theoretical grounding for tempering the posterior under model misspecification, but applies a single temperature to the entire likelihood. Our analysis motivates a two-parameter rescaling that also works well in practice.

2. Bayesian Last Layer for MLIPs

We estimate uncertainty by placing a Bayesian linear model on the final force-prediction layer of an MLIP, which is often a graph neural network (Schütt et al., 2017) or a transformer (Liao et al., 2024), while treating the rest of the network as a deterministic feature extractor. Let ϕ_θ denote the backbone and penultimate force-head representation. For an atomic configuration with n_a atoms, the model produces per-atom features $\phi_i \in \mathbb{R}^D$, which are mapped to Cartesian forces in $\mathbb{R}^3$. Since n_a varies across structures such as molecules or unit cells, we take atoms rather than structures as the unit of Bayesian inference; therefore, a dataset of structures is flattened into N atoms, producing a fixed-dimensional regression problem with features $\Phi \in \mathbb{R}^{N\times D}$ and force targets $Y \in \mathbb{R}^{N\times K}$, where $K = 3$; we discuss mini-batching later.

The final linear force head is modelled as

$$Y = \Phi B + E, \qquad B \in \mathbb{R}^{D\times K}. \tag{1}$$

This per-atom formulation avoids padding or structure-level aggregation and gives a single posterior over B that can be used for structures of arbitrary size. The same last-layer weights are shared across all atoms; information about the parent structure enters through the node-level features produced by the backbone. For the Bayesian treatment, we need to specify the form of the likelihood and the prior.

Likelihood. We use a *matrix-normal* likelihood,

$$Y \mid B, \Phi, \Sigma_e \sim \mathcal{MN}(\Phi B,\ I_N,\ \Sigma_e), \tag{2}$$

or, equivalently, $\mathrm{vec}(Y) \mid B, \Phi, \Sigma_e \sim \mathcal{N}(\mathrm{vec}(\Phi B),\ \Sigma_e \otimes I_N)$, where $\mathrm{vec}(\cdot)$ stacks the columns of its matrix argument. This is the same as assuming $\varepsilon_i \sim \mathcal{N}(0, \Sigma_e)$ independently across atomic environments. The identity row covariance is an important approximation: conditional on B and the atomic features Φ, the residual force errors of different atomic observations are modelled as independent, including atoms belonging to the same structure.[1]

[1]It does not imply that the model's predicted forces are

Prior. We use the conjugate matrix-normal prior

$$B \mid \Lambda_0, \Sigma_e \sim \mathcal{MN}(0,\ \Lambda_0^{-1},\ \Sigma_e), \tag{4}$$

where $\Lambda_0 = \mathrm{diag}(\lambda_1, \ldots, \lambda_D)$ is the prior precision over the features. Sharing the column covariance Σ_e between the likelihood and prior is the standard conjugate construction that results in a closed-form posterior. The diagonal precision provides a scalable feature-wise shrinkage prior. A dense $D \times D$ prior covariance would introduce substantially more parameters and more fragile inversions, while the diagonal form is sufficient to regularize poorly constrained feature directions and to enter the marginal likelihood through the Bayesian complexity penalty. However, our main rationale for using a diagonal form is the fact that due to the massive size of the dataset, likelihood dominates and the role of prior is mainly reduced to regularize extreme feature directions.

Posterior. Combining (2) and (4), the posterior over the last-layer weights is

$$B \mid Y, \Phi, \Lambda_0, \Sigma_e \sim \mathcal{MN}(B_n,\ \Lambda_n^{-1},\ \Sigma_e),$$

with

$$\Lambda_n = \Phi^\top \Phi + \Lambda_0, \qquad B_n = \Lambda_n^{-1} \Phi^\top Y. \tag{5}$$

Marginal likelihood. We estimate Λ_0 and Σ_e by empirical Bayes. The marginal likelihood is the density of the observed force targets under the likelihood-prior model after removing the unobserved last-layer weights:

$$p(Y \mid \Phi, \Lambda_0, \Sigma_e) = \int p(Y \mid B, \Phi, \Sigma_e)\, p(B \mid \Lambda_0, \Sigma_e)\, \mathrm{d}B. \tag{6}$$

Here $p(\cdot)$ denotes the densities associated with the likelihood and prior in Equations (2) and (4). Thus, B is integrated out of the joint density $p(Y, B \mid \Phi, \Lambda_0, \Sigma_e)$. Evaluating this Gaussian integral and taking the negative logarithm gives

$$\begin{aligned} \mathcal{L}_{\mathrm{NLML}} = & \frac{NK}{2} \log(2\pi) + \frac{N}{2} \log |\Sigma_e| \\ & + \frac{K}{2} \left(\log |\Lambda_n| - \log |\Lambda_0| \right) \\ & + \frac{1}{2} \mathrm{tr} \left[\Sigma_e^{-1} \left(R^\top R + B_n^\top \Lambda_0 B_n \right) \right], \\ & \text{where } R = Y - \Phi B_n \end{aligned} \tag{7}$$

marginally independent after Bayesian inference. Once B is integrated out, predictions at two atoms with features ϕ_i and ϕ_j become correlated through the shared posterior uncertainty in B:

$$\mathrm{Cov}(y_i, y_j \mid Y, \Phi) = \delta_{ij} \Sigma_e + \left(\phi_i^\top \Lambda_n^{-1} \phi_j \right) \Sigma_e. \tag{3}$$

.

Equation (7) is differentiable in the hyperparameters and, through Φ, in the parameters of the feature extractor. We therefore use it to learn the last-layer prior and noise model, and optionally to fine-tune the representation.

At the scale of modern MLIP datasets, recomputing full-dataset sufficient statistics at every optimization step is infeasible. During training, we optimize minibatch estimates of (7), using the structures in each mini-batch to form $\Phi^\top \Phi$ and $\Phi^\top Y$. When features are fixed (i.e., the base model is unchanged), we calculate the full covariance once and rescale it appropriately for a given batch size in the objective above. In this setting, our optimization remains exact. After training, we recompute the posterior sufficient statistics on the full training set and use the resulting full-data posterior for test-time prediction.

Posterior Predictive. For a test atom with feature vector $\phi_* \in \mathbb{R}^D$, the posterior predictive distribution is

$$\begin{aligned} y_* \mid \phi_*, Y, \Phi \sim \mathcal{N}\left(\phi_*^\top B_n,\ \Sigma_e + u(\phi_*) \Sigma_e \right), \\ u(\phi_*) = \phi_*^\top \Lambda_n^{-1} \phi_*. \end{aligned} \tag{8}$$

3. Post-hoc Covariance Rescaling

The predictive covariance in Equation (8) decomposes into a constant term Σ_e and an input-dependent term $u(\phi_*)\Sigma_e$. This decomposition is exact under two assumptions of the Bayesian linear model:

(A1) *Linearity.* Given the encoder ϕ_θ, there exist weights B^* such that the noiseless target satisfies $y = \phi_\theta(x)^\top B^*$.

(A2) *Gaussian noise.* Observed targets are corrupted by additive noise $\varepsilon \sim \mathcal{N}(0, \Sigma_e^*)$.

Both assumptions are violated in the MLIP setting. (A2) fails because, conditional on an atomic configuration and a fixed DFT protocol, the target force is deterministic, so $\Sigma_e^* = 0$. (A1) fails because the linear span of ϕ_θ does not contain the exact target force field. Two distinct mechanisms produce this failure:

(M1) *In-distribution misspecification.* A single linear head over ϕ_θ does not fit the targets on the training set; the training residuals are nonzero.

(M2) *Feature collapse on out-of-distribution inputs.* Distinct OOD configurations with different targets are mapped onto similar training features by ϕ_θ, so no linear head over the fixed encoder can fit them simultaneously.

The two mechanisms have different implications for what post-hoc recalibration can correct; Sections 3.1 to 3.3 develop this distinction. Throughout, we avoid the labels

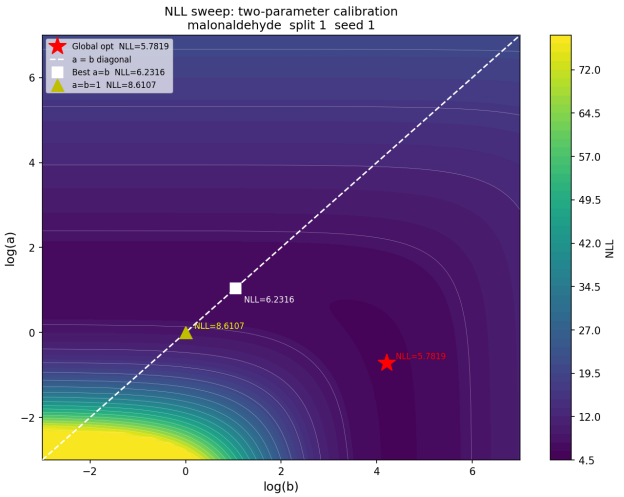

Figure 1. **Validation NLL surface for the decoupled rescaling on Malonaldehyde.** Validation NLL as a function of the rescaling parameters (a, b) from Equation (12); darker contours indicate better calibration. The dashed line marks the single-temperature subspace $a = b$ to which any scalar tempering of the BLL covariance is restricted. The unconstrained optimum (red star) sits well off the diagonal and improves substantially over both the raw BLL covariance (yellow triangle) and the best single-temperature solution (white square). This asymmetry matches our analysis: the last-layer posterior $u(\phi_*)\Sigma_e$ is too narrow and requires large upward rescaling (Proposition 3.1), while the residual floor Σ_e is much closer to the correct scale. Absolute NLL values are elevated by a heavy right tail of high-error validation points.

"aleatoric" and "epistemic" for the two covariance components, since neither retains its standard meaning under (A1)–(A2) violations. We refer to Σ_e as the *residual floor* and to $u(\phi_*)\Sigma_e$ as the *last-layer posterior covariance*.

3.1. Misspecification of the Residual Covariance

We first characterize Σ_e under empirical Bayes. The part of the NLML in Equation (7) depending on Σ_e is

$$\mathcal{L}(\Sigma_e) = \frac{N}{2}\log|\Sigma_e| + \frac{1}{2}\operatorname{tr}\left[\Sigma_e^{-1}\left(R^\top R + B_n^\top \Lambda_0 B_n\right)\right] \tag{9}$$

where each row of R is a per-atom force residual after the linear fit. Setting $\partial\mathcal{L}/\partial\Sigma_e^{-1} = 0$ yields the stationary point

$$\widehat{\Sigma}_e = \tfrac{1}{N}R^\top R + \tfrac{1}{N}B_n^\top \Lambda_0 B_n. \tag{10}$$

The first term is the empirical second moment of the training residuals; the second is the prior shrinkage contribution, which is small once $N \gg D$. Empirical Bayes therefore sets Σ_e to (approximately) the residual covariance of the regularized fit.

Since DFT forces are deterministic given the configuration and protocol, Y contains no stochastic component. The fitted predictions are $\Phi B_n = HY$, where $H = \Phi(\Phi^\top\Phi + \Lambda_0)^{-1}\Phi^\top$ is the regularized hat matrix and the residuals are $R = (I - H)Y$; substituting in Equation (10), we get

$$\widehat{\Sigma}_e = \tfrac{1}{N}Y^\top(I-H)^\top(I-H)Y + \tfrac{1}{N}B_n^\top\Lambda_0 B_n. \tag{11}$$

This means that Σ_e does not estimate noise; under (A2) failure, there is none to estimate, and $\widehat{\Sigma}_e$ instead absorbs the in-distribution misspecification (M1). A calibrated residual floor remains useful in the deterministic setting, since it improves predictive likelihood. However, the empirical-Bayes value of Σ_e is unlikely to be appropriately scaled for predictive scoring on held-out data, for two reasons. First, NLML evaluates Σ_e against training residuals and the marginal-likelihood penalty rather than against held-out predictive likelihood. Second, as established in Section 3.2, the same Σ_e enters the input-dependent term through the conjugate prior, so its NLML value reflects a compromise between two distinct roles. The post-hoc scalar a introduced in Section 3.3 corrects the floor; b corrects the input-dependent term independently.

3.2. The Last-Layer Posterior under a Fixed Encoder

The input-dependent term $u(\phi_*)\Sigma_e$ from Equation (8), with $u(\phi_*) = \phi_*^\top\Lambda_n^{-1}\phi_*$, is the predictive variance contributed by the posterior over B. This posterior places mass only on the function class $\{f_B(x) = \phi_\theta(x)^\top B : B \in \mathbb{R}^{D\times K}\}$ obtained by varying B over the fixed encoder.

Substituting $\Lambda_n = \Phi^\top\Phi + \Lambda_0$, we have $u(\phi_*) = \phi_*^\top(\Phi^\top\Phi + \Lambda_0)^{-1}\phi_*$, which is the squared Mahalanobis distance of ϕ_* from the training feature distribution up to the regularizer. It is small for ϕ_* in directions of high training mass and large for ϕ_* in directions weakly covered by training. The qualitative content of this quantity is geometric and does not depend on (A1) or (A2): a generic predictor is less reliable on inputs whose features lie far from the training support, since fewer data constrain the local fit. The ranking that $u(\phi_*)$ induces over test inputs is therefore robust to (M1) and to (A2) failure.

Robustness fails for the magnitude of $u(\phi_*)\Sigma_e$. The matrix scaling the input-dependent term is the same Σ_e that appears in the noise covariance, since the conjugate prior $B \sim \mathcal{MN}(0, \Lambda_0^{-1}, \Sigma_e)$ ties the two together to admit a closed-form posterior. As a result, Σ_e simultaneously sets the residual floor, where empirical Bayes inflates it to absorb the (M1) residuals (Section 3.1), and scales the input-dependent term. There is no reason these two roles require the same magnitude. The post-hoc scalar b introduced in Section 3.3 corrects the magnitude of the input-dependent term without altering its geometric ranking, and Section 3.4 shows that this correction is equivalent to tempering the conditional posterior over B.

Mechanism (M2) lies outside what scalar rescaling can cor-

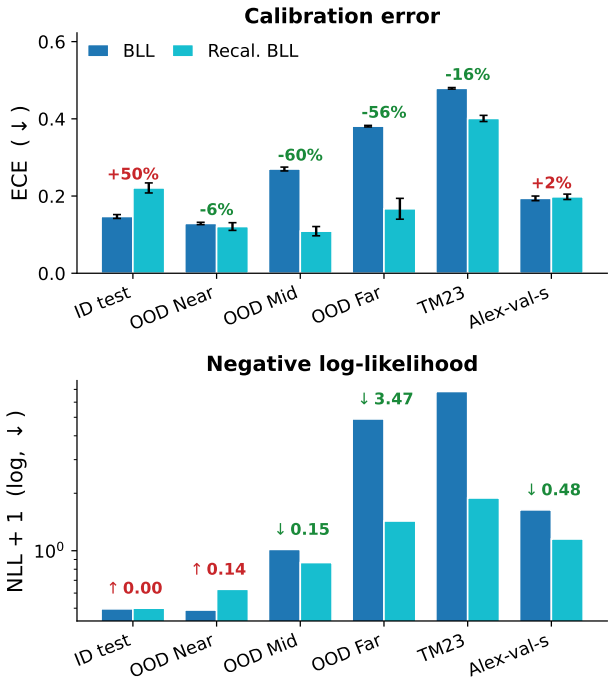

Figure 2. **Recalibration improves both calibration and likelihood across most splits.** Top: expected calibration error (ECE) before and after two-parameter scaling on the BLL predictive variance. Bottom: negative log-likelihood (NLL+1 on a log scale, since raw NLL can be negative). Annotations show the relative change. Recalibration delivers large ECE reductions on the harder OOD splits (-56% to -60%) and a $\sim$3.5 unit NLL drop on OOD Far, at the cost of slight over-correction on the well-calibrated ID test set. MAE and Spearman correlation are unchanged by recalibration and reported in Table 2 and Figure 3.

rect. When the encoder collapses OOD inputs onto well-supported feature regions, $u(\phi_*)$ is small on those inputs even though the predictions are unreliable, so the ranking itself is incorrect; no choice of b recovers it. Mechanism (M2) therefore remains a residual source of error after rescaling that we cannot address within our framework.

3.3. Decoupled Covariance Rescaling

The raw BLL covariance from Equation (8) couples the two components through Σ_e, $\mathrm{Cov}(y_* \mid \phi_*, Y, \Phi) = (1 + u(\phi_*))\Sigma_e$. A single scalar T would rescale this to $T(1 + u(\phi_*))\Sigma_e$, preserving the ratio between the floor and the input-dependent component. As established in Sections 3.1 and 3.2, however, the empirical-Bayes value of Σ_e reflects a compromise between two roles whose appropriate predictive magnitudes need not coincide. We therefore consider the two-parameter rescaling

$$\mathrm{Cov}_{a,b}(y_* \mid \phi_*, Y, \Phi) = a\,\Sigma_e + b\,u(\phi_*)\Sigma_e, \qquad a, b > 0, \tag{12}$$

in which a rescales the residual floor and b rescales the input-dependent term independently. Section 3.4 shows that b corresponds to a temperature on the conditional posterior over B. A single global scalar is restricted to the diagonal $a = b$ and cannot reach the optimum when it lies off the diagonal; Figure 1 shows the validation-NLL surface for a representative case.

We estimate (a, b) on a held-out in-distribution validation set by minimizing the predictive negative log-likelihood,

$$(a^\star, b^\star) = \arg\min_{a,b>0} \sum_{i \in \mathcal{D}_{\mathrm{val}}} -\log \mathcal{N}\left(y_i;\, \phi_i^\top B_n,\, \Sigma_i(a,b)\right), \tag{13}$$

where the adjusted covariance is defined as

$$\Sigma_i(a,b) = a\Sigma_e + b\,u(\phi_i)\Sigma_e.$$

optimizing the unconstrained parameterization $\alpha, \beta \in \mathbb{R}$ with $a = e^\alpha$, $b = e^\beta$ to enforce positivity. The posterior mean is unaffected; only the predictive covariance is recalibrated.

3.4. Connection to Posterior Tempering

The scalar b admits a direct interpretation as a temperature on the conditional posterior over B. Consider tempering the unnormalized joint density by a factor $w > 0$,

$$p_w(B \mid Y, \Phi) \propto \left[p(Y \mid B, \Phi, \Sigma_e)\, p(B \mid \Lambda_0, \Sigma_e)\right]^w, \tag{14}$$

which parameterizes a controlled widening or narrowing of the posterior over B.

Proposition 3.1 (Tempering rescales the last-layer covariance)**.** *Tempering the joint density in Equation (14) by w changes the posterior precision from Λ_n to $\Lambda_{n,w} = w\Lambda_n$, and consequently $u_w(\phi_*) = \phi_*^\top \Lambda_{n,w}^{-1} \phi_* = w^{-1} u(\phi_*)$. Multiplying $u(\phi_*)\Sigma_e$ by b is therefore equivalent to a temperature $T = b = 1/w$ on the conditional posterior of B.*

Proof. The terms in the unnormalized log joint quadratic in B are, up to constants, $-\frac{1}{2}\,\mathrm{tr}\left[\Sigma_e^{-1} B^\top (\Phi^\top \Phi + \Lambda_0) B\right]$. Multiplying the log joint by w scales $(\Phi^\top \Phi + \Lambda_0)$ by w, so the tempered posterior precision is $\Lambda_{n,w} = w(\Phi^\top \Phi + \Lambda_0) = w\Lambda_n$. The linear-in-$B$ terms also scale by w, so the posterior mean is unchanged: $B_{n,w} = (w\Lambda_n)^{-1}(w\Phi^\top Y) = B_n$. Substituting yields $u_w(\phi_*) = w^{-1} u(\phi_*)$. □

Values $b > 1$ correspond to a warmer conditional posterior over B and inflate the input-dependent uncertainty; $b < 1$ corresponds to a cooler posterior. The required temperature is determined post hoc because the spread of B needed for calibrated prediction depends on the extent to which the empirical-Bayes value of Σ_e was inflated to absorb the (M1) residuals.

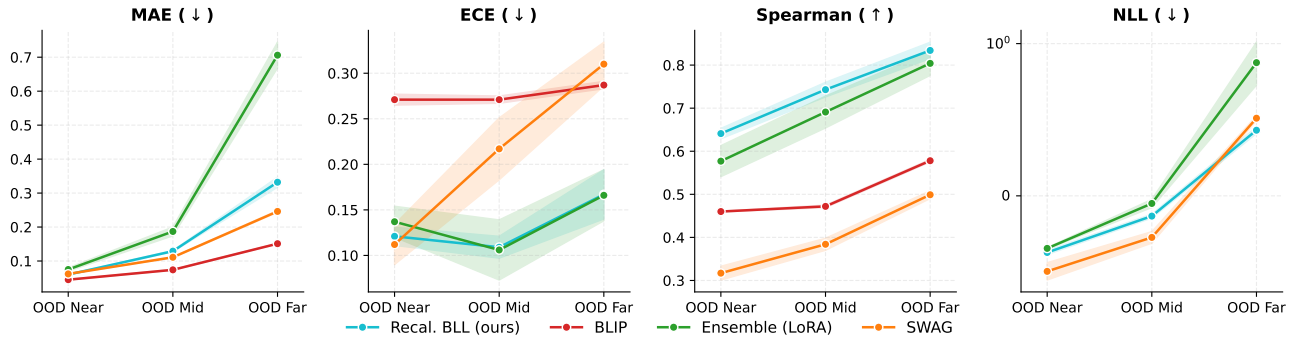

Figure 3. **Performance across increasing distribution shift (OOD Near → OOD Mid → OOD Far).** Each panel reports a different metric; shaded bands show ±1 standard deviation over seeds. Recalibrated BLL maintains the strongest Spearman correlation across all OOD splits and remains competitive on ECE and NLL, while keeping MAE growth substantially below the ensemble. BLIP is omitted from the NLL panel because its values blow up on OOD Mid and OOD Far.

4. Experiments

We evaluate BLL on two benchmarks chosen to span the deployment regimes of foundation MLIPs. MPtrj is the The Materials Project's trajectory dataset (Deng, 2023). We train on this dataset and evaluate on a chemically heterogeneous in-distribution split, as well as a distribution shift by stratifying held-out structures by their energy above the relaxed state into OOD-Near, OOD-Mid, and OOD-Far, plus two further transfer sets (the transition-metal benchmark TM23 (Owen et al., 2024) and a non-WBM Alexandria subsample (Barroso-Luque et al., 2024), Alex-val-s). rMD17 dataset (Christensen & lilienfeld, 2020) evaluates a different regime; the model is trained on a single molecule's trajectory and asked to extrapolate to a held-out split of the same molecule (ID) and to all non-training molecules (OOD), a setting in which a linear head over a frozen encoder is most likely to overfit. Across both benchmarks, we compare BLL — in its raw and post-hoc-recalibrated forms — against the LoRA deep ensemble, variational-dropout BLIP, and SWAG (Maddox et al., 2019), with exact Gaussian processes added on rMD17, and report force MAE, expected calibration error (ECE), and the Spearman correlation between predicted uncertainty and absolute force error. The recalibration parameters $(a^\star, b^\star)$ are fit only on an in-distribution validation set (Equation (13)); improvements on more distorted splits, therefore, reflect OOD generalization of the calibration rather than tuning on the test distribution. For further details, refer to Appendix B and C.

Table 1. Train and inference cost.

Method	Train (A100-hr)	Inf. (passes)
BLL	10.5	1
Ensemble (LoRA)	19.63	3
BLIP	26.5*	10
SWAG	10.8	10

4.1. Calibration, ranking, and rescaling across distribution shift

Figure 3 tracks BLL performance against the LoRA ensemble, BLIP, and SWAG across the three increasingly distorted MPtrj OOD splits. Recalibrated BLL maintains the strongest Spearman correlation throughout, with its ranking power growing as distortion increases. On ECE and NLL, recalibrated BLL tracks the ensemble within overlapping error bands, while MAE growth stays well below it (0.332 vs. 0.706 at OOD-Far). The pattern carries through to the additional splits in Table 2: recalibrated BLL achieves the highest Spearman on four of the five (MPtrj ID, MPtrj TM23, rMD17 ID, rMD17 OOD), with Alex-val-s the lone exception (BLL 0.764 vs. BLIP 0.781). The rMD17 OOD split is the most pronounced instance of the pattern: although BLL has the worst MAE there (0.259), since a single-molecule training trajectory provides little extrapolation signal for the linear head, its Spearman of 0.821 is the second largest gap to the next-best method (0.702 for the ensemble) of any split in our comparison. Recalibration does not rescue OOD ECE in this regime (0.483 → 0.463), a boundary case we examine in Section 5.

Figure 2 isolates the contribution of the post-hoc rescaling. The rescaling produces 56–60% ECE reductions on OOD-Mid and OOD-Far and a ∼3.5-unit NLL improvement on OOD-Far, at the cost of mild over-correction on the already well-calibrated ID test split. The rescaling parameters $(a^\star, b^\star)$ are fit only on an in-distribution validation split (eq. 13); improvements on TM23, Alex-val-s, OOD-Mid, and OOD-Far therefore reflect generalization of the calibration rather than fitting on the test distribution. The trade-off favours BLL by a wide margin in deployment terms: a small ECE penalty on near-distribution data buys order-of-magnitude calibration gains exactly where overconfident predictions would otherwise be most damaging in MD.

Table 2. Model performance on MPTrj and rmd17 data splits. Best results for each split are highlighted in bold. (↓ indicates lower is better; ↑ indicates higher is better).

Data Split	Model	MAE (↓)	ECE (↓)	Spearman (↑)	NLL (↓)
MPtrj ID test	Recalibrated BLL (ours)	0.038 ± 0.000	0.221 ± 0.013	**0.603** ± 0.009	-0.500 ± 0.018
	BLIP	**0.029** ± 0.001	0.268 ± 0.010	0.504 ± 0.001	-
	Ensemble (LoRA)	0.046 ± 0.002	0.267 ± 0.024	0.579 ± 0.024	-0.514 ± 0.013
	SWAG	0.043 ± 0.001	**0.117** ± 0.023	0.329 ± 0.014	**-0.521** ± 0.062
MPtrj TM23 test	Recalibrated BLL (ours)	0.542 ± 0.029	0.401 ± 0.008	**0.866** ± 0.027	**0.889** ± 0.052
	BLIP	**0.195** ± 0.004	**0.400** ± 0.004	0.718 ± 0.014	-
	Ensemble (LoRA)	0.780 ± 0.022	0.403 ± 0.021	0.664 ± 0.029	1.647 ± 0.221
	SWAG	0.361 ± 0.026	0.474 ± 0.001	0.644 ± 0.007	1.060 ± 0.186
MPtrj Alex-val-s test	Recalibrated BLL (ours)	0.077 ± 0.001	0.198 ± 0.007	0.764 ± 0.015	0.152 ± 0.014
	BLIP	**0.052** ± 0.000	0.162 ± 0.003	**0.781** ± 0.003	-
	Ensemble (LoRA)	0.106 ± 0.022	0.224 ± 0.001	0.709 ± 0.025	**-0.062** ± 0.028
	SWAG	0.077 ± 0.001	**0.129** ± 0.007	0.366 ± 0.015	0.247 ± 0.017
rmd17 ID test	Recalibrated BLL (ours)	**0.030** ± 0.015	0.132 ± 0.067	**0.611** ± 0.056	-0.922 ± 0.181
	GP-rbf	0.057 ± 0.019	**0.101** ± 0.042	0.538 ± 0.58	13.11 ± 28.79
	GP-5/2	0.056 ± 0.019	0.116 ± 0.017	0.555 ± 0.024	**-1.321** ± 0.338
	Ensemble (LoRA)	0.098 ± 0.019	0.152 ± 0.039	**0.611** ± 0.234	0.445 ± 0.109
rmd17 OOD test	Recalibrated BLL (ours)	0.259 ± 0.045	0.463 ± 0.017	**0.821** ± 0.079	0.278 ± 0.200
	GP-rbf	0.177 ± 0.056	0.245 ± 0.092	0.581 ± 0.125	11.67 ± 28.28
	GP-5/2	0.166 ± 0.050	**0.199** ± 0.097	0.550 ± 0.131	0.876 ± 1.169
	Ensemble (LoRA)	**0.125** ± 0.011	0.260 ± 0.045	0.702 ± 0.071	**-0.356** ± 0.074

4.2. Computational cost

Table 1 reports training cost in A100 GPU-hours per fully trained instance and inference cost as backbone forward passes per prediction. BLL is the cheapest method on both axes: training takes $\sim$10.5 hours — comparable to a single SWAG run and roughly half the cost of the K=3 LoRA ensemble ($\sim$19.6 hours total) — and inference requires a single backbone pass, against 3 for the ensemble and 10 each for BLIP and SWAG. The only additional cost is a one-time recomputation of the posterior sufficient statistics on the full training set after backbone training, which is negligible relative to training itself. BLIP was trained on RTX 8000 GPUs and its reported training time (*) is hardware-adjusted using a speedup factor measured from matched LoRA-ensemble runs on both architectures.

5. Discussion

Why the rescaling works, and when it does not. The two scalars play different roles. The floor multiplier a requires only a small adjustment, averaging 0.476 across MPtrj splits, since empirical Bayes already inflates Σ_e to absorb the training residuals (Section 3.1). The input-dependent multiplier b does the substantive work, taking values in the range 10^5 to 10^6. This is because with N in the millions, Λ_n grows linearly in N, so $u(\phi_*)$ shrinks as $1/N$. The role of b is to undo this contraction, which by Proposition 3.1 is equivalent to tempering the posterior over B.

Parameters fit on an in-distribution validation set transfer well to OOD inputs on MPtrj. The validation mean squared Mahalanobis distance moves from 24.2 to 3.84 (target 3.0), and the same $(a^\star, b^\star)$ reduces OOD-Far from 675.8 to 10.5 and its ECE from 0.39 to 0.20. The recalibration generalizes because b corrects the $1/N$ shrinkage, which is shared across splits. The procedure fails on rMD17, where OOD ECE moves only from 0.483 to 0.463. The cause is that rMD17 trains and validates on a single molecule's trajectory, leaving too little variability to identify $(a^\star, b^\star)$. The requirement is not that the validation set resemble the deployment distribution, but that it be sufficiently diverse — a condition foundation-MLIP datasets meet easily.

Ranking and calibration as separate axes. BLL achieves the highest Spearman correlation on nearly every split, including rMD17 OOD where its MAE and ECE are poor. ECE measures whether predicted variances match the magnitudes of errors; Spearman measures whether they rank inputs correctly. The post-hoc scalars act uniformly on $u(\phi_*)$ and cannot alter the ranking, so the Spearman performance comes from $u(\phi_*)$ itself — a Mahalanobis distance from the test feature to the training distribution that depends only on the frozen backbone.

Limitations. Compute constraints prevented us from running full deep ensembles; our ensemble comparator approximates them via LoRA fine-tuning of a shared backbone with $K{=}3$ members, both of which may understate the calibration quality of larger ensembles of independently trained models. The BLL formulation we present targets a non-equivariant, direct force-prediction head: extending it to conservative architectures (forces obtained as energy gradients) and to equivariant backbones such as NequIP or MACE raises structural questions about the last-layer parameterization that we do not address here. Our experiments use a single foundation backbone (orb-v3), so whether the geometric ranking we exploit transfers cleanly to other architectures remains open. Finally, the active-learning use case motivated in the introduction is supported here only through the proxy of strong Spearman correlation; an end-to-end demonstration in an MD-driven labelling loop is the natural next validation.

6. Conclusion

Bayesian last-layer inference is a practical alternative to deep ensembles for foundation-scale neural force fields, providing a closed-form posterior on a single fixed backbone at negligible inference cost. Applied directly, however, it is overconfident, for two distinct reasons. Empirical Bayes inflates Σ_e to absorb training residuals, since the targets are deterministic and there is no genuine noise to estimate. At the same time, the posterior over B contracts as $1/N$, so the input-dependent term vanishes at large dataset sizes. The two effects act independently and require independent correction, which we provide via a two-parameter rescaling fit on a standard validation split.

Empirically, the recalibrated BLL matches or exceeds deep ensembles on ECE across all six MPtrj splits, while requiring roughly half the training time and a single backbone pass at inference. It also produces the strongest ranking of uncertainty against error on nearly every split. Two limitations of calibrated BLL are that: 1) post-hoc rescaling cannot correct for feature collapse, and 2) the recalibration requires a validation set with sufficient diversity, the lack of which is the principal source of error on rMD17. Overall, our findings suggest that BLL is a practical UQ method for force fields, and it is worth investigating its applications to downstream tasks, such as molecular dynamics and active learning.

Impact statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Barroso-Luque, L., Shuaibi, M., Fu, X., Wood, B. M., Dzamba, M., Gao, M., Rizvi, A., Zitnick, C. L., and Ulissi, Z. W. Open materials 2024 (omat24) inorganic materials dataset and models, 2024. URL https://arxiv.org/abs/2410.12771.

Bartók, A. P., Payne, M. C., Kondor, R., and Csányi, G. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, 104(13):136403, 2010.

Bartók, A. P., Kondor, R., and Csányi, G. On representing chemical environments. *Physical Review B*, 87(18): 184115, 2013.

Bissiri, P. G., Holmes, C. C., and Walker, S. G. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, 2016.

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

Choudhary, K. et al. Ai-driven expansion and application of the alexandria database. *arXiv preprint arXiv:2512.09169*, 2025/2026. URL https://doi.org/10.48550/arXiv.2512.09169.

Christensen, A. S. and lilienfeld, A. V. Revised MD17 dataset (rMD17). 7 2020. doi: 10.6084/m9.figshare.12672038.v4. URL https://figshare.com/articles/dataset/Revised_MD17_dataset_rMD17_/12672038.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace Redux – effortless Bayesian deep learning. In *Advances in Neural Information Processing Systems*, 2021.

Deng, B. Materials Project Trajectory (MPtrj) Dataset. 7 2023. doi: 10.6084/m9.figshare.23713842.v2. URL https://figshare.com/articles/dataset/Materials_Project_Trjectory_MPtrj_Dataset/23713842.

Fu, X., Wood, B. M., Barroso-Luque, L., Levine, D. S., Gao, M., Dzamba, M., and Zitnick, C. L. Learning smooth and expressive interatomic potentials for physical property prediction, 2025. URL https://arxiv.org/abs/2502.12147.

Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, 2011.

Grünwald, P. and van Ommen, T. Inconsistency of bayesian inference for misspecified linear models, and a proposal for repairing it, 2018. URL https://arxiv.org/abs/1412.3730.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. What are Bayesian neural network posteriors really like? In *International Conference on Machine Learning*, 2021.

Kuleshov, V., Fenner, N., and Ermon, S. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, 2018.

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

Liao, Y.-L., Wood, B., Das, A., and Smidt, T. EquiformerV2: Improved equivariant transformer for scaling to higher-degree representations. In *International Conference on Learning Representations*, 2024.

Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Advances in Neural Information Processing Systems*, 2020.

MacKay, D. J. C. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3): 448–472, 1992.

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning, 2019. URL https://arxiv.org/abs/1902.02476.

Neal, R. M. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, 1996.

Owen, C., Torrisi, S., Xie, Y., Batzner, S., Bystrom, K., Coulter, J., Musaelian, A., Sun, L., and Kozinsky, B. Complexity of many-body interactions in transition metals via machine-learned force fields from the tm23 data set. *npj Computational Materials*, 10(1):105, May 2024. doi: 10.1038/s41524-024-01264-z. URL https://www.nature.com/articles/s41524-024-01264-z.

Papamarkou, T., Skoularidou, M., Palla, K., Aitchison, L., Arbel, J., Dunson, D., Filippone, M., Fortuin, V., Hennig, P., Hernández-Lobato, J. M., et al. Position: Bayesian deep learning is needed in the age of large-scale AI. In *International Conference on Machine Learning*, 2024.

Pearce, T., Leibfried, F., and Brintrup, A. Uncertainty in neural networks: Approximately Bayesian ensembling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Pilania, G., Gubernatis, J. E., and Lookman, T. Multi-fidelity machine learning models for accurate bandgap predictions of solids. *Computational Materials Science*, 129:156–163, 2017.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Rhodes, B., Vandenhaute, S., Šimkus, V., Gin, J., Godwin, J., Duignan, T., and Neumann, M. Orb-v3: atomistic simulation at scale, 2025. URL https://arxiv.org/abs/2504.06231.

Ritter, H., Botev, A., and Barber, D. A scalable Laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Schütt, K. T., Kindermans, P.-J., Sauceda, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, 2017.

Tan, A. R., Urata, S., Goldman, S., Dietschreit, J. C. B., and Gómez-Bombarelli, R. Single-model uncertainty quantification in neural network potentials does not consistently outperform model ensembles. *Npj Comput. Mater.*, 9(1), December 2023.

Thaler, S., Doehner, G., and Zavadlav, J. Scalable Bayesian uncertainty quantification for neural network potentials: Promise and pitfalls. *Journal of Chemical Theory and Computation*, 2023.

van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, 2020.

van Amersfoort, J., Smith, L., Jesson, A., Key, O., and Gal, Y. On feature collapse and deep kernel learning for single forward pass uncertainty. In *arXiv preprint arXiv:2102.11409*, 2021.

Watson, J., Lin, J. A., Klink, P., Pajarinen, J., and Peters, J. Latent derivative Bayesian last layer networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*, 2020.

Wood, B. M., Dzamba, M., Fu, X., Gao, M., Shuaibi, M., Barroso-Luque, L., Abdelmaqsoud, K., Gharakhanyan, V., Kitchin, J. R., Levine, D. S., Michel, K., Sriram, A., Cohen, T., Das, A., Rizvi, A., Sahoo, S. J., Ulissi, Z. W., and Zitnick, C. L. Uma: A family of universal models for atoms, 2025. URL https://arxiv.org/abs/2506.23971.

Algorithm 1 Bayesian Last Layer for MLIPs: training, calibration, and prediction.

Require: Pretrained backbone ϕ_θ; train/val/test sets $\mathcal{D}_{\text{tr}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{te}}$

procedure TRAIN($\phi_\theta, \mathcal{D}_{\text{tr}}$)
  Initialize hyperparameters Λ_0, Σ_e (and optionally θ)
  for each minibatch $\mathcal{B} \subset \mathcal{D}_{\text{tr}}$ **do**
    Compute per-atom features $\Phi_\mathcal{B}$ via ϕ_θ and stack force targets $Y_\mathcal{B}$
    Form $\Phi_\mathcal{B}^\top \Phi_\mathcal{B}$ and $\Phi_\mathcal{B}^\top Y_\mathcal{B}$; compute Λ_n, B_n via (5)
    Evaluate $\mathcal{L}_{\text{NLML}}$ on $\mathcal{B}$ via (7)
    Update $\{\Lambda_0, \Sigma_e\}$ (and optionally θ) by gradient descent on $\mathcal{L}_{\text{NLML}}$
  end for
  Recompute $\Phi^\top \Phi$, $\Phi^\top Y$ on the full $\mathcal{D}_{\text{tr}}$
  Compute final posterior (Λ_n, B_n) via (5)
  return $\theta, \Lambda_0, \Sigma_e, \Lambda_n, B_n$
end procedure

procedure CALIBRATE($\theta, \Lambda_n, B_n, \Sigma_e, \mathcal{D}_{\text{val}}$)
  for each $(x_i, y_i) \in \mathcal{D}_{\text{val}}$ **do**
    Compute $\phi_i = \phi_\theta(x_i)$ and $u(\phi_i) = \phi_i^\top \Lambda_n^{-1} \phi_i$
  end for
  Solve (13) for $(a^\star, b^\star)$ via gradient descent on $(\alpha, \beta) = (\log a, \log b)$
  return $(a^\star, b^\star)$
end procedure

procedure PREDICT($x_*; \theta, B_n, \Lambda_n, \Sigma_e, a^\star, b^\star$)
  $\phi_* \leftarrow \phi_\theta(x_*)$; $u(\phi_*) \leftarrow \phi_*^\top \Lambda_n^{-1} \phi_*$
  Mean: $\hat{y}_* \leftarrow \phi_*^\top B_n$ {from (8)}
  Covariance: $\widehat{\text{Cov}}(y_*) \leftarrow a^\star \, \Sigma_e + b^\star \, u(\phi_*) \, \Sigma_e$ {from (12)}
  return $(\hat{y}_*, \widehat{\text{Cov}}(y_*))$
end procedure=0

A. Derivation of empirical bayes with learned global noise

A.1. initial setup

We use a Bayesian Linear Last Layer to perform Bayesian regression on embeddings from Orb-v3 (Rhodes et al., 2025) to atomic forces, allowing for direct force prediction as well as uncertainty estimations for these predictions.
Multivariate Regression Setup:

$$Y = \Phi B + E \tag{15}$$

$$E \sim \mathcal{N}(0, \Sigma_e) \tag{16}$$

Where $Y \in \mathbb{R}^{N \times K}$ are the targets, $\Phi \in \mathbb{R}^{N \times D}$ are the features from the network, $B \in \mathbb{R}^{D \times K}$ are the weights of the final layer and $E \in \mathbb{R}^{N \times K}$ is the noise. N is the number of data points, K is the output dimension (for forces in XYZ K is 3) and D is the dimension of the feature embeddings.
We assume each row $y_n \in \mathbb{R}^{1 \times K}$ is independently distributed as

$$y_n \mid B, \Sigma_e \sim \mathcal{N}(\phi_n B, \Sigma_e) \tag{17}$$

So the joint likelihood over all rows follows a Matrix Normal Distribution.

$$p(Y \mid B, \Phi, \Sigma_e) = \mathcal{MN}(Y \mid \Phi B, \; I_N, \; \Sigma_e) \tag{18}$$

Note on the Matrix Normal: If $X \sim \mathcal{MN}(M, U, V)$ with $X \in \mathbb{R}^{n \times p}$, $U \in \mathbb{R}^{n \times n}$ (row covariance), $V \in \mathbb{R}^{p \times p}$ (column covariance), the density is:

$$(2\pi)^{-np/2} |V|^{-n/2} |U|^{-p/2} \exp\Big(-\frac{1}{2} \text{tr}[V^{-1}(X - M)^T U^{-1}(X - M)]\Big) \tag{19}$$

And so the density can be written as:

$$p(Y \mid B, \Phi, \Sigma_e) = \frac{1}{(2\pi)^{NK/2}\,|\Sigma_e|^{N/2}} \exp\Big(-\tfrac{1}{2}\ \mathrm{tr}\big[\Sigma_e^{-1}(Y-\Phi B)^T(Y-\Phi B)\big]\Big) \tag{20}$$

A.2. conjugate Matrix Normal prior on B

We place a Matrix Normal prior on B with zero mean, a diagonal row precision Λ_0 and column covariance Σ_e (same as noise for conjugacy)

$$B \mid \Lambda_0, \Sigma_e \sim \mathcal{MN}(0, \Lambda_0^{-1}, \Sigma_e) \tag{21}$$

So

$$p(B \mid \Lambda_0, \Sigma_e) = \frac{|\Lambda_0|^{K/2}}{(2\pi)^{DK/2}\,|\Sigma_e|^{D/2}} \exp\Big(-\tfrac{1}{2}\ \mathrm{tr}\big[\Sigma_e^{-1} B^T \Lambda_0\, B\big]\Big) \tag{22}$$

A.3. marginal likelihood and integrating out B

The Marginal Likelihood is given by:

$$p(Y \mid \Phi, \Lambda_0, \Sigma_e) = \int p(Y \mid B, \Phi, \Sigma_e)\, p(B \mid \Lambda_0, \Sigma_e)\ \mathrm{d}B \tag{23}$$

$$= \int \frac{1}{(2\pi)^{NK/2}\,|\Sigma_e|^{N/2}} \exp\Big(-\tfrac{1}{2}\ \mathrm{tr}\big[\Sigma_e^{-1}(Y-\Phi B)^T(Y-\Phi B)\big]\Big) * \frac{|\Lambda_0|^{K/2}}{(2\pi)^{DK/2}\,|\Sigma_e|^{D/2}} \exp\Big(-\tfrac{1}{2}\ \mathrm{tr}\big[\Sigma_e^{-1} B^T \Lambda_0\, B\big]\Big)$$

$$= \int \frac{|\Lambda_0|^{K/2}}{(2\pi)^{(N+D)K/2}\,|\Sigma_e|^{(N+D)/2}} \exp\Big(-\tfrac{1}{2}\ \mathrm{tr}\Big[\Sigma_e^{-1}((Y-\Phi B)^T(Y-\Phi B) + B^T\Lambda_0\, B)\Big]\Big)$$

The argument of the exponential is:

$$-\frac{1}{2}\,\mathrm{tr}\Big[\Sigma_e^{-1}(\underbrace{(Y-\Phi B)^T(Y-\Phi B)}_{\text{likelihood}} + \underbrace{B^T\Lambda_0\, B}_{\text{prior}})\Big]$$

We need to integrate out B, so first we expand and collect the terms in B. First we expand the likelihood term:

$$(Y-\Phi B)^T(Y-\Phi B) = Y^TY - Y^T\Phi\, B - B^T\Phi^TY + B^T\Phi^T\Phi\, B$$

Add the prior term back in:

$$Q(B) = Y^TY - Y^T\Phi\, B - B^T\Phi^TY + B^T\big(\Phi^T\Phi + \Lambda_0\big)B$$

Define the posterior precision

$$\Lambda_n = \Phi^T\Phi + \Lambda_0 \tag{24}$$

and the posterior mean given by the standard MLE solution

$$B_n = \Lambda_n^{-1}\Phi^TY \tag{25}$$

$$\implies \Lambda_n B_n = \Phi^TY \tag{26}$$

Substituting into the terms of $Q(B)$:

$$Y^TY - B_n^T\Lambda_n^T B - B^T\Lambda_n B_n + B^T\Lambda_n B$$

$$= Y^TY + (B - B_n)^T\Lambda_n(B - B_n) - B_n^T\Lambda_n B_n$$

$$Q(B) = \underbrace{Y^TY - B_n^T\Lambda_n B_n}_{\text{constant w.r.t. } B} + \underbrace{(B - B_n)^T\Lambda_n(B - B_n)}_{\text{integrates out}} \tag{27}$$

We only integrate over the $(B - B_n)^T\Lambda_n(B - B_n)$ term since other terms are constant w.r.t. B

$$\int \exp\Big(-\tfrac{1}{2}\operatorname{tr}\big[\Sigma_e^{-1}(B - B_n)^T\Lambda_n(B - B_n)\big]\Big)\,\mathrm{d}B$$

This is the unnormalized integral of a $\mathcal{MN}(B_n, \Lambda_n^{-1}, \Sigma_e)$ density which evaluates to:

$$(2\pi)^{DK/2}\,|\Sigma_e|^{D/2}\,|\Lambda_n^{-1}|^{K/2} = (2\pi)^{DK/2}\,|\Sigma_e|^{D/2}\,|\Lambda_n|^{-K/2}$$

A.4. assembling marginal likelihood

$$p(Y \mid \Phi, \Lambda_0, \Sigma_e) = \underbrace{\frac{1}{(2\pi)^{NK/2}|\Sigma_e|^{N/2}}}_{\text{from likelihood}} \times \underbrace{\frac{|\Lambda_0|^{K/2}}{(2\pi)^{DK/2}|\Sigma_e|^{D/2}}}_{\text{from prior}} \times \underbrace{(2\pi)^{DK/2}|\Sigma_e|^{D/2}|\Lambda_n|^{-K/2}}_{\text{from integral}}$$

$$\times\ \exp\Big(-\tfrac{1}{2}\operatorname{tr}\big[\Sigma_e^{-1}(Y^TY - B_n^T\Lambda_n B_n)\big]\Big)$$

After canceling out, we are left with:

$$\boxed{p(Y \mid \Phi, \Lambda_0, \Sigma_e) = \frac{|\Lambda_0|^{K/2}}{(2\pi)^{NK/2}\,|\Sigma_e|^{N/2}\,|\Lambda_n|^{K/2}}\ \exp\Big(-\tfrac{1}{2}\operatorname{tr}\big[\Sigma_e^{-1}(Y^TY - B_n^T\Lambda_n B_n)\big]\Big)} \tag{28}$$

Taking the $-\log$ we get our Negative Log Marginal Likelihood which can be used for our loss function:

$$\mathcal{L} = \frac{NK}{2}\log(2\pi) + \frac{N}{2}\log|\Sigma_e| + \frac{K}{2}\big(\log|\Lambda_n| - \log|\Lambda_0|\big) + \frac{1}{2}\operatorname{tr}\big[\Sigma_e^{-1}(Y^TY - B_n^T\Lambda_n B_n)\big] \tag{29}$$

The derivation could end here but we further rewrite the loss for better interpretability.

A.5. decomposing loss to improve interpretability

We can rewrite the data-dependent term in two forms,
Form A:

$$\operatorname{tr}\big[\Sigma_e^{-1}(Y^TY - B_n^T\Lambda_n B_n)\big]$$

Form B (residuals + prior penalty):

$$(Y - \Phi B_n)^T(Y - \Phi B_n) = Y^TY - Y^T\Phi B_n - B_n^T\Phi^TY + B_n^T\Phi^T\Phi B_n$$

$$= Y^TY - B_n^T\Lambda_n B_n - B_n^T\Lambda_n B_n + B_n^T(\Lambda_n - \Lambda_0)B_n$$

$$= \underbrace{Y^T Y - B_n^T \Lambda_n B_n}_{\text{Form A}} - B_n^T \Lambda_0 B_n$$

A.6. Final NLML

For gradient descent, the constant $\frac{NK}{2} \log 2\pi$ can be dropped, however, we keep it so we can report the NLML.

$$\boxed{\mathcal{L} = \frac{NK}{2} \log 2\pi + \frac{N}{2} \log |\Sigma_e| + \frac{K}{2} \big(\log |\Lambda_n| - \log |\Lambda_0| \big) + \frac{1}{2} \operatorname{tr} \big[\Sigma_e^{-1} \big((Y - \Phi B_n)^T (Y - \Phi B_n) + B_n^T \Lambda_0 B_n \big) \big]} \quad (30)$$

The final loss only depends on Σ_e and Λ_0, so these are the two parameters we learn using gradient descent during training. Additionally, you can also learn the parameters of the feature model simultaneously.

B. Experiments

All deep models share the orb-direct-20 backbone (https://github.com/orbital-materials/orb-models) pretrained on OMat24, released under the Apache 2.0 license (https://www.apache.org/licenses/LICENSE-2.0). Deep methods use a batch size of 64; the exact GP baselines on rMD17 use closed-form inference and do not require batched training.

For rMD17, we train only on ethanol, malonaldehyde, toluene, uracil, and benzene — the five smallest molecules — since the remaining trajectories are too large for tractable exact GP regression. OOD evaluation is performed across all non-training molecules in the rMD17 dataset.

Training BLL. We remove the final projection layer from the orb-v3 force head and use the resulting penultimate embeddings as the BLL features ϕ. The backbone is frozen during training; only the penultimate force projection and the priors Λ_0, Σ_e are updated by gradient descent on the NLML objective (eq. 7). We initialize Σ_e and Λ_0 as identity matrices and optimize with Adam at learning rate 1×10^{-3}, with early-stopping patience 10 epochs. ZBL forces (as computed by the Orb model) are subtracted from the targets during training; on rMD17 we additionally convert units to eV/Å. We report results averaged over 5 seeds on MPtrj and 3 seeds per training molecule on rMD17. The recalibration parameters $(a^\star, b^\star)$ are fit via Nelder–Mead on the same validation split used for early stopping. MPtrj training typically converges in $\sim$70 epochs at $\sim$130 seconds per epoch on 4 A100s; rMD17 epochs run in a few seconds on an RTX 8000 but typically require 200–300 epochs to converge — its total compute is a small fraction of MPtrj's.

Training Ensemble LoRA. Each ensemble member reinitializes the last projection layer of the force head and adds a variance head that predicts log-variances. The variance head copies the force-head architecture with all weights and biases set to zero, so that the initial predicted variance is unity. We inject LoRA adapters (rank 8, $\alpha = 16$) into the message-passing MLPs. Training minimizes Gaussian NLL using Adam with learning rate 1×10^{-3} for the LoRA matrices and 5×10^{-4} for the force and variance heads, with early-stopping patience 10 epochs. On MPtrj we train 9 members (seeds 0–8) and form three ensembles of 3 members each ($\{0,1,2\}, \{3,4,5\}, \{6,7,8\}$); on rMD17 we train one ensemble of 3 members per training molecule (seeds 0, 1, 2). MPtrj members converged in $\sim$73 epochs at $\sim$184 seconds per epoch on 4 A100s; rMD17 training, as with BLL, accounts for a small fraction of total compute.

Training BLIP. We use the training script provided by the authors (https://github.com/dario-coscia/blip) with default hyperparameters and an added early-stopping patience of 10 epochs. MPtrj results are averaged over seeds 1, 2, 3. BLIP was tested only on MPtrj.

Training SWAG. We first train the orb-v3 force head on MSE using Adam at learning rate 1×10^{-3}, with the backbone frozen. After 10 epochs without improvement we reload the best checkpoint and continue training for 40 additional epochs at the same learning rate. SWAG retains the last 20 weight deltas to construct the low-rank covariance and averages over all 40 SWAG-collection epochs to estimate the mean. Reported numbers are averaged over seeds 0, 1, 2, with 10 posterior samples drawn at inference. SWAG was tested only on MPtrj.

Computational resources. Experiments were distributed across NVIDIA A100 (80GB), RTX 8000 (48GB), and L40S (48GB) GPUs, using 4 GPUs per training run with 8–16 CPUs per GPU and 256GB of system memory. Per-method training costs on a single 4-A100 instance are reported in Table 1; BLIP was trained on RTX 8000 hardware, and its A100-equivalent figure was obtained by scaling its measured wall-clock by a factor of 3.51, derived from 2 matched LoRA-ensemble runs across both architectures. No single seed exceeded 48 hours of wall-clock. Including failed runs, hyperparameter exploration, and the recalibration sweep, total project compute is approximately 375 A100-equivalent GPU-hours (9 LoRA members $\times$ 20 hr + 5 BLL seeds $\times$ 10 hr + 3 SWAG seeds $\times$ 10 hr + 3 BLIP seeds $\times$ 30 hr, plus rMD17 runs).

C. Datasets

MPtrj. The Materials Project trajectory dataset (Deng, 2023) (https://doi.org/10.6084/m9.figshare.23713842; MIT license, https://opensource.org/license/MIT). We partition MPtrj by per-atom force above relaxed-state energy, producing an in-distribution split (< 0.005 eV/Å) and three increasingly distorted out-of-distribution splits at $[0.005, 0.0175)$, $[0.0175, 0.070)$, and $[0.070, \infty)$ eV/Å, denoted OOD-Near, OOD-Mid, and OOD-Far. We split MP-IDs 80/10/10 (training/validation/test) using random seed 1.

rMD17. The revised MD17 dataset (Christensen & lilienfeld, 2020) (https://doi.org/10.6084/m9.figshare.12672038; CC0 1.0, https://creativecommons.org/publicdomain/zero/1.0/). We use the predefined index-1 train/test splits for all molecules.

TM23. The TM23 transition-metal benchmark (Owen et al., 2024) (CC BY-NC-ND 4.0, https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en; dataset URL: https://archive.materialscloud.org/records/tcrks-ymp88/files/benchmarking_master_collection-20240316T202423Z-001.zip). We evaluate on the entire dataset (both train and test splits combined) without modification, consistent with the license's no-derivatives clause, and use it strictly for academic evaluation.

Alex-val-s. The Alexandria validation subsample distributed with OMat24 (Barroso-Luque et al., 2024; Choudhary et al., 2025/2026) (https://huggingface.co/datasets/facebook/OMAT24; CC BY 4.0, https://creativecommons.org/licenses/by/4.0/legalcode). We use the entire dataset for evaluation.