Scalable Derivative Gaussian Processes via Exact Gradient Reduction

Hyunseok Seung
Department of Statistics
University of Wisconsin–Madison
Madison, WI, 53706
hseung2@wisc.edu

Matthias Katzfuss
Department of Statistics
University of Wisconsin–Madison
Madison, WI, 53706
katzfuss@gmail.com

Abstract

Gradient observations can substantially improve Gaussian process (GP) surrogates, particularly in high-dimensional settings where function evaluations are expensive. However, exact inference with n function values and n full gradients in d dimensions scales cubically in the joint state size, imposing an intractable $\mathcal{O}(n^3d^3)$ computational bottleneck. We introduce TERA, a highly scalable derivative GP method that combines target-specific exact gradient reduction with a Vecchia approximation. We prove that for stationary kernels under exact or metric-matched noisy gradients, the gradient components orthogonal to the directions connecting the target and conditioning points are conditionally independent of the target function value; consequently, the exact conditional density is fully characterized by at most m^2 directional derivatives once a conditioning set of size m is specified. By using these reduced, dimension-free conditionals as local factors in a Vecchia approximation, TERA effectively decouples n and d from the dense matrix inversion. This reduces the per-target evaluation cost to $\mathcal{O}(dm^2+m^6)$ time and $\mathcal{O}(dm^2+m^4)$ memory, preserving the corresponding full-gradient target conditional. Empirical evaluations demonstrate that TERA achieves state-of-the-art predictive accuracy while operating orders of magnitude faster than standard derivative GPs. Crucially, both computation time and peak GPU memory remain essentially flat with respect to d over the tested range, enabling highly scalable inference in high-dimensional spaces. Code is available at https://github.com/hseung88/tera.

1 Introduction

Gradient observations are increasingly common in surrogate modeling, especially when data are generated by computational models equipped with automatic differentiation [4, 2] or adjoint derivative solvers [16]. The downstream target, however, is often not the gradient field itself. Gaussian-process (GP) regression requires calibrated prediction of a function response [25], and Bayesian optimization (BO) requires a surrogate that can locate extrema of a scalar objective [34]. Each query may therefore provide a full d-dimensional gradient, while the surrogate is ultimately used through function-value predictions and decisions.

Derivative GPs provide a principled model for such data by coupling function values and input derivatives through kernel derivatives [29, 25]. Exact inference with n function values and n full gradients in d dimensions involves a Gaussian state of size $n(d+1)$, which makes dense linear algebra and memory costs prohibitive when either n or d is moderate or large. Existing scalable derivative GP methods reduce this cost through kernel interpolation [11, 14], inducing derivative variables [23], and structured derivative covariance decompositions [10]. These methods reduce the cost of inference in derivative GPs while keeping the joint function-gradient state as the main computational object.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

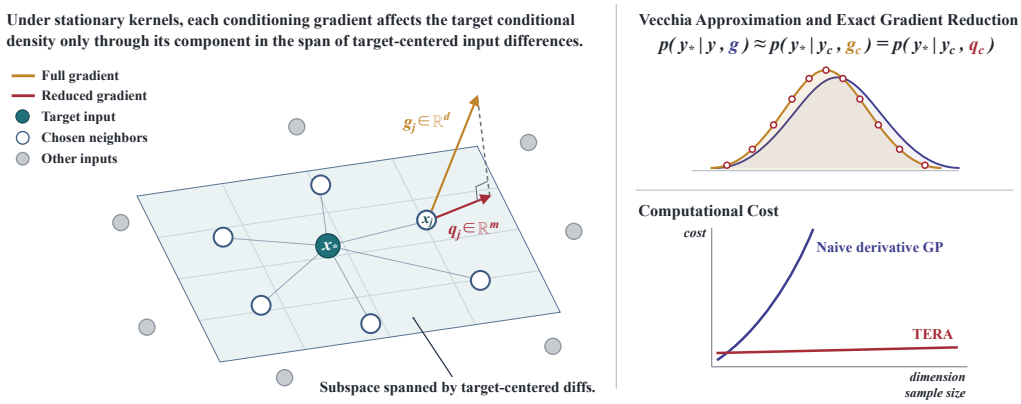

Figure 1: TERA predicts a function value using observations at m neighboring inputs. Within each Vecchia conditional factor, the full local gradient $\mathbf{g}_c \in \mathbb{R}^{md}$ is replaced by $\mathbf{q}_c \in \mathbb{R}^{m^2}$, formed from directional derivatives along target-centered input differences. For stationary kernels under certain gradient-noise conditions, this preserves the full-gradient conditional density, leaving the Vecchia approximation as the only gap relative to exact derivative GP inference.

We instead start from function-value prediction and use the target to determine how the observed gradients enter the relevant conditional distributions. For a function-value target, after nearby function observations have been conditioned on, the observed gradients affect the target only through their contribution to its conditional mean and conditional variance. Preserving these two quantities is sufficient for the target prediction. For a stationary prior kernel, the relevant contribution is not tied to the original d gradient coordinates. It is carried by linear functionals of observed gradients (i.e., directional derivatives) along directions spanned by differences between the target input and the inputs where gradients are observed, with the kernel metric determining the geometry. This reduction applies to any fixed conditioning set and reduces the derivative information needed for one scalar target, but it does not by itself control how many gradient locations enter the conditioning set.

We control the number of conditioning locations by combining the reduction with Vecchia approximations [32, 30, 9, 20, 19, 26, 18]. A target function value $f_\star$ is conditioned on m neighboring locations rather than all observations, and the gradient reduction is applied within that conditioning set. The full gradients in the resulting Vecchia factor have dimension md, while the reduction for stationary kernels replaces them by at most m^2 target-specific gradient statistics (Figure 1). This replacement gives the same conditional Gaussian distribution as the one obtained by conditioning on the full gradients in that set. These reduced conditional factors are used both for prediction and for evaluating the conditional Vecchia likelihood used to learn kernel and noise parameters. With m conditioning locations and $m \ll d$, each factor can be evaluated in $\mathcal{O}(dm^2 + m^6)$ time, with only linear dependence on d from reduced local block assembly. For likelihood optimization, minibatching over Vecchia factors gives stochastic updates with cost $\mathcal{O}(|\mathcal{B}|(dm^2 + m^6))$ for a minibatch $\mathcal{B}$, removing the explicit dependence on n from each update when $|\mathcal{B}|$ is fixed. We refer to the resulting method as TERA, for Target-specific Exact gradient Reduction for Accelerating derivative GP inference.

The paper makes the following contributions.

- We prove an exact target-specific reduction for derivative GP conditionals. For stationary kernels, we show that a target function value conditioned on m reference inputs depends on their md gradient coordinates only through at most m^2 reduced gradients, under exact or metric-matched noisy gradient observations.
- We turn this reduction into a scalable Vecchia derivative GP method by constructing reduced covariance blocks directly from target-centered Gram matrices. This yields per-factor cost $\mathcal{O}(dm^2 + m^6)$ and supports a conditional Vecchia learning objective.
- We show empirically that TERA matches the posterior accuracy of the corresponding full-gradient Vecchia derivative GP in fidelity tests and uses gradients for better function-value prediction and lower BO regret at costs comparable to standard GP baselines.

2 Related Work

Scalable derivative GP inference. Recent scalable derivative GP methods reduce the cost of using derivative observations by changing the global covariance computation or posterior representation. Kernel interpolation methods, including D-SKI [11] and DSoftKI [14], accelerate derivative covariance computations through structured or learned interpolation, but their accuracy and fitting cost remain tied to a global interpolation approximation. Inducing variable methods such as DSVGP and DDSVGP build sparse variational posterior approximations using derivative or directional derivative inducing variables [23], with directional derivatives reducing high-dimensional cost at the price of an additional directional approximation. In contrast, we focus on settings where gradients are available but the inferential target is function-value prediction. In such settings, TERA expresses the gradient dependence of each chosen target conditional density through low-dimensional exact statistics, without introducing a global interpolation approximation or a variational inducing representation.

Stationary kernel structures. The closest algebraic connection is de Roos et al. [10]. For stationary kernels, their decomposition expresses the gradient covariance matrix as the sum of a Kronecker product term and a low-rank correction, improving the scalability of exact derivative GP inference with respect to d at the cost of poor scalability with respect to n ($\mathcal{O}(n^6)$ time). In contrast, we use stationarity to identify the directions through which the conditioning gradients can affect a chosen target function value. When applied inside Vecchia factors, the reduction makes the derivative solve depend on the conditioning set size m, rather than the total number of observations n, achieving scalability to both large n and large d.

Vecchia approximations for scalable GP inference. Vecchia approximations have been used in scalable GP inference [6, 7] and BO [17]. In existing methods, conditioning sets are chosen to approximate dependence among function-value observations. A direct derivative GP analogue would still condition on all gradient coordinates from the chosen inputs. We instead use each Vecchia factor with an exact lower-dimensional representation of the full gradient for the corresponding function-value prediction.

3 Preliminaries

3.1 Derivative GP Observation Model

We place a GP prior $f(\cdot) \sim \mathcal{GP}(\mu_{\boldsymbol{\theta}}(\cdot), k_{\boldsymbol{\theta}}(\cdot,\cdot))$ on $\mathcal{D} \subset \mathbb{R}^d$ and assume that $k_{\boldsymbol{\theta}}$ is twice continuously differentiable in its input arguments. For training inputs $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n\times d}$, write $f_i = f(\mathbf{x}_i)$ and observe $y_i = f_i + \epsilon_i^y$ and $\mathbf{g}_i = \nabla_{\mathbf{x}} f(\mathbf{x}_i) + \boldsymbol{\epsilon}_i^g$. The observation errors are independent across inputs and independent of the GP, with $\epsilon_i^y \sim \mathcal{N}(0, \sigma_y^2)$ and $\boldsymbol{\epsilon}_i^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \mathbf{I}_d)$. We stack the observations as $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$ and $\mathbf{g} = (\mathbf{g}_1^\top, \dots, \mathbf{g}_n^\top)^\top \in \mathbb{R}^{nd}$.

The pair $(\mathbf{y}, \mathbf{g})$ is jointly Gaussian,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{g} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_y \\ \boldsymbol{\mu}_g \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{yy} & \mathbf{K}_{yg} \\ \mathbf{K}_{gy} & \mathbf{K}_{gg} \end{bmatrix} \right),$$

where $\mathbf{K}_{yy} = \mathbf{K}_{ff} + \sigma_y^2 \mathbf{I}_n$ and $[\mathbf{K}_{ff}]_{ij} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)$. The derivative covariance blocks are understood blockwise as

$$[\mathbf{K}_{yg}]_{i,j} = \nabla_{\mathbf{x}_j} k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)^\top, \qquad [\mathbf{K}_{gg}]_{i,j} = \nabla_{\mathbf{x}_i} \nabla_{\mathbf{x}_j}^\top k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) + \mathbb{1}\{i=j\}\sigma_g^2 \mathbf{I}_d.$$

Exact gradient observations correspond to $\sigma_g^2 = 0$.

For a prediction input $\mathbf{x}_\star$, write $f_\star = f(\mathbf{x}_\star)$ and $\mathbf{z} = (\mathbf{y}^\top, \mathbf{g}^\top)^\top$. Let $\boldsymbol{\mu}_z$ and $\mathbf{K}_{zz}$ denote the mean and covariance of $\mathbf{z}$, and let $\mathbf{k}_{f_\star z} = \mathbf{Cov}(f_\star, \mathbf{z})$. If $\mathbf{K}_{zz} \succ 0$, the GP predictive mean and variance are

$$\begin{aligned} \mu_{f_\star|\mathbf{z}} &= \mu_{\boldsymbol{\theta}}(\mathbf{x}_\star) + \mathbf{k}_{f_\star z} \mathbf{K}_{zz}^{-1} (\mathbf{z} - \boldsymbol{\mu}_z), \\ \sigma_{f_\star|\mathbf{z}}^2 &= k_{\boldsymbol{\theta}}(\mathbf{x}_\star, \mathbf{x}_\star) - \mathbf{k}_{f_\star z} \mathbf{K}_{zz}^{-1} \mathbf{k}_{z f_\star}, \end{aligned}$$

where $\mathbf{k}_{z f_\star} = \mathbf{k}_{f_\star z}^\top$. When predicting a noisy future observation, σ_y^2 is added to $\sigma_{f_\star|\mathbf{z}}^2$.

3.2 Stationary Kernels and Target-Centered Differences

We use stationary kernels of the form

$$k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = \kappa_{\boldsymbol{\theta}}\left((\mathbf{x}-\mathbf{x}')^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}}(\mathbf{x}-\mathbf{x}')\right) = \kappa_{\boldsymbol{\theta}}(\|\tilde{\mathbf{x}}-\tilde{\mathbf{x}}'\|^2), \tag{1}$$

where $\boldsymbol{\Lambda}_{\boldsymbol{\theta}} \succ 0$ is symmetric positive definite and includes isotropic and ARD metrics [25], and $\tilde{\mathbf{x}} = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{1/2}\mathbf{x}$ denotes the scaled input.

For a target input $\mathbf{x}_i$ and reference inputs $\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_m}$, define

$$\mathbf{D}_i = [\mathbf{x}_{j_1} - \mathbf{x}_i, \ldots, \mathbf{x}_{j_m} - \mathbf{x}_i] \in \mathbb{R}^{d\times m}, \qquad \boldsymbol{\Delta}_i = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{1/2}\mathbf{D}_i = [\tilde{\mathbf{x}}_{j_1} - \tilde{\mathbf{x}}_i, \ldots, \tilde{\mathbf{x}}_{j_m} - \tilde{\mathbf{x}}_i].$$

The corresponding Gram matrix is

$$\mathbf{H}_i = \boldsymbol{\Delta}_i^\top \boldsymbol{\Delta}_i = \mathbf{D}_i^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}} \mathbf{D}_i, \qquad [\mathbf{H}_i]_{ab} = (\mathbf{x}_{j_a} - \mathbf{x}_i)^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}}(\mathbf{x}_{j_b} - \mathbf{x}_i).$$

Thus $\boldsymbol{\Delta}_i$ contains the target-centered differences in scaled input coordinates. The matrix $\mathbf{H}_i$ contains their inner products, from which all pairwise squared distances among the target and reference inputs can be obtained. For example, $\|\tilde{\mathbf{x}}_{j_a} - \tilde{\mathbf{x}}_{j_b}\|^2 = [\mathbf{H}_i]_{aa} + [\mathbf{H}_i]_{bb} - 2[\mathbf{H}_i]_{ab}$.

3.3 Vecchia Ordering and Conditioning Sets

Vecchia approximations [32, 19] replace full chain-rule conditioning in an ordered Gaussian model by small conditioning sets. For an ordered vector $\mathbf{y} = (y_1, \ldots, y_n)^\top$,

$$p(\mathbf{y}) = \prod_{i=1}^{n} p(y_i \mid y_1, \ldots, y_{i-1}) \approx \prod_{i=1}^{n} p\big(y_i \mid \mathbf{y}_{c(i)}\big),$$

where $c(i) \subset \{1, \ldots, i-1\}$ and $|c(i)| \le m$. We use a maximum-minimum-distance (maximin) ordering and take $c(i)$ to be the m_i nearest previously ordered neighbors of $\mathbf{x}_i$ in the $\boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{1/2}$-scaled input space, where $m_i = |c(i)|$, following scaled Vecchia constructions [21].

4 Methods

4.1 Targeted Gradient Reduction for Function Values

We first state the reduction for a generic finite reference set. The result does not depend on how the reference set is chosen and will later be applied to Vecchia conditioning sets for prediction and parameter estimation.

Fix a target input $\mathbf{x}_i \in \mathbb{R}^d$ and a reference set $c = \{j_1, \ldots, j_{m_c}\}$ with $i \notin c$. Let $\mathbf{y}_c = (y_{j_1}, \ldots, y_{j_{m_c}})^\top$ and $\mathbf{g}_c = (\mathbf{g}_{j_1}^\top, \ldots, \mathbf{g}_{j_{m_c}}^\top)^\top \in \mathbb{R}^{m_c d}$. Let $\mathbf{D}_i \in \mathbb{R}^{d\times m_c}$ be the target-centered difference matrix from Section 3.2 for the reference inputs in c. Define the target-specific reduced gradient statistic $\mathbf{q}_c$ as

$$\mathbf{q}_c = (\mathbf{I}_{m_c} \otimes \mathbf{D}_i^\top)\mathbf{g}_c = \begin{bmatrix} \mathbf{D}_i^\top \mathbf{g}_{j_1} \\ \vdots \\ \mathbf{D}_i^\top \mathbf{g}_{j_{m_c}} \end{bmatrix} \in \mathbb{R}^{m_c^2}, \qquad \mathbf{D}_i^\top \mathbf{g}_{j_a} = \begin{bmatrix} (\mathbf{x}_{j_1} - \mathbf{x}_i)^\top \mathbf{g}_{j_a} \\ \vdots \\ (\mathbf{x}_{j_{m_c}} - \mathbf{x}_i)^\top \mathbf{g}_{j_a} \end{bmatrix} \in \mathbb{R}^{m_c}. \tag{2}$$

The entries of $\mathbf{q}_c$ can be viewed as unnormalized directional derivatives. Each term $(\mathbf{x}_{j_b} - \mathbf{x}_i)^\top \mathbf{g}_{j_a}$ is the inner product between the observed gradient at $\mathbf{x}_{j_a}$ and the target-centered difference $\mathbf{x}_{j_b} - \mathbf{x}_i$. These directions are determined by the chosen target input and reference set, rather than selected as global derivative directions. When the target input and reference set are clear, we refer to $\mathbf{q}_c$ simply as the reduced gradient. The reduction is target-specific because $\mathbf{D}_i$ depends on the target input $\mathbf{x}_i$.

The basic object used by TERA is the conditional density $p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c)$. The next lemma gives its Schur-complement form. All proofs can be found in Appendix A.

Lemma 1 (Conditional density with reduced gradients)**.** *Under the joint Gaussian model for $(y_i, \mathbf{y}_c, \mathbf{q}_c)$, assume that the joint covariance matrix is positive definite. Then*

$$p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c) = \mathcal{N}\left(\mu_{y_i|\mathbf{y}_c} + t_i,\, \sigma^2_{y_i|\mathbf{y}_c} - s_i\right), \tag{3}$$

where

$$t_i = \mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c}\mathbf{K}^{-1}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}\left(\mathbf{q}_c - \boldsymbol{\mu}_{\mathbf{q}_c|\mathbf{y}_c}\right), \qquad s_i = \mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c}\mathbf{K}^{-1}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}\mathbf{k}_{\mathbf{q}_c y_i|\mathbf{y}_c}.$$

Here $\mu_{y_i|\mathbf{y}_c}$, $\sigma^2_{y_i|\mathbf{y}_c}$, $\boldsymbol{\mu}_{\mathbf{q}_c|\mathbf{y}_c}$, $\mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c}$, and $\mathbf{K}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}$ are Gaussian conditional moments after conditioning on $\mathbf{y}_c$.

The terms t_i and s_i are the mean correction and variance reduction induced by conditioning on the reduced gradient. Under the assumptions stated below, we will show that this reduced gradient preserves the same conditional density obtained by conditioning on the full reference gradients. This lets us evaluate the same target conditional density using reduced covariance blocks (whose size does not depend on d) rather than forming and factorizing full derivative covariance blocks.

Proposition 1 (Targeted gradient reduction)**.** *Assume the stationary kernel model in* (1)*. Suppose that the observed gradients are either exact, so that $\sigma_g^2 = 0$, or have independent Gaussian noise matched to the scaled input metric, $\boldsymbol{\epsilon}_i^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2\boldsymbol{\Lambda}_{\boldsymbol{\theta}})$. For any target input $\mathbf{x}_i$ and any reference set $c = \{j_1, \ldots, j_{m_c}\}$ with $i \notin c$,*

$$p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{g}_c) = p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c). \tag{4}$$

Thus the full reference gradient vector in $\mathbb{R}^{m_c d}$ can be replaced by the reduced gradient $\mathbf{q}_c \in \mathbb{R}^{m_c^2}$ without changing the conditional density for the chosen function-value target.

The proof is given in Appendix A.2, and it uses the target-centered differences defined in Section 3.2. The matched noise condition is automatically satisfied by exact gradients and by isotropic kernels under the iid unscaled gradient noise model in Section 3.1. For other noise models, conditioning on the reduced gradient still defines a valid Gaussian conditional density, but exact equivalence to full noisy gradient conditioning need not hold. Appendix B discusses extensions of the reduction to dot-product kernels and a class of nonstationary kernels defined through input or feature transformations. Such transformations are used in input warping [27] and deep kernel learning [33].

Direct Construction and Scalability. The primary advantage of replacing the full reference gradient $\mathbf{g}_c$ with the reduced gradient $\mathbf{q}_c$ is computational. Evaluating the conditional density $p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c)$ requires the conditional blocks in Lemma 1. A naive approach would form the full derivative covariance of $\mathbf{g}_c$, an $m_c d \times m_c d$ matrix, and then apply the linear map in (2). This route requires storing $\mathcal{O}(m_c^2 d^2)$ covariance entries, which can be prohibitive for large d. Instead, we construct the reduced blocks directly from the stationary kernel geometry.

Lemma 2 (Direct construction of reduced blocks)**.** *Assume the kernel and gradient-noise conditions of Proposition 1. For a target input $\mathbf{x}_i$ and reference set c, the reduced blocks $\boldsymbol{\mu}_{\mathbf{q}_c|\mathbf{y}_c}$, $\mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c}$, and $\mathbf{K}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}$ needed in Lemma 1 can be computed directly from the elements of the $m_c \times m_c$ target-centered Gram matrix $\mathbf{H}_i = \mathbf{D}_i^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}} \mathbf{D}_i$. This bypasses the instantiation of the full $m_c d \times m_c d$ derivative covariance, yielding the $m_c^2 \times m_c^2$ reduced blocks directly.*

The explicit algebraic formulas for these blocks are provided in Appendix A.3. Crucially, the dense Cholesky factorization of the resulting reduced conditional covariance $\mathbf{K}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}$ costs $\mathcal{O}(m_c^6)$. This eliminates the cubic dependence on the input dimension d from the full derivative computation, allowing our method to scale to high-dimensional problems where standard derivative GPs can fail.

4.2 Conditional Vecchia Likelihood with Reduced Gradients

We now apply the target-specific reduction from Section 4.1 to the Vecchia conditioning sets from Section 3.3. For the ith training factor, write $c(i) = \{j_1, \ldots, j_{m_i}\}$ with $m_i = |c(i)|$. Let $\mathbf{D}_i \in \mathbb{R}^{d \times m_i}$ be the target-centered difference matrix formed from $\mathbf{x}_i$ and the ordered inputs in $c(i)$. The reduced gradient vector for the ith factor is $\mathbf{q}_{c(i)} = (\mathbf{I}_{m_i} \otimes \mathbf{D}_i^\top)\mathbf{g}_{c(i)} \in \mathbb{R}^{m_i^2}$. This is the specialization of $\mathbf{q}_c$ in Section 4.1 to the Vecchia set $c(i)$. Define the conditional Vecchia density with reduced gradients as

$$\tilde{p}^{\mathrm{Vec}}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{g}) = \prod_{i=1}^{n} p_{\boldsymbol{\theta}}\left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)}\right). \tag{5}$$

Each factor is a normalized Gaussian density in y_i. Since $c(i) \subset \{1, \ldots, i-1\}$, sequential integration over $y_n, \ldots, y_1$ shows that (5) is a normalized conditional density over $\mathbf{y}$ for fixed observed gradients.

We estimate hyperparameters by maximizing

$$\tilde{\ell}_n(\boldsymbol{\theta}) = \log \tilde{p}^{\mathrm{Vec}}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{g}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right). \tag{6}$$

The parameter vector $\boldsymbol{\theta}$ includes the kernel parameters and the noise levels σ_y^2 and σ_g^2. For fixed conditioning sets, the reduced gradients $\mathbf{q}_{c(i)}$ are deterministic functions of the observed gradients and inputs. The Gaussian covariance blocks used to evaluate each factor depend on $\boldsymbol{\theta}$. By applying Lemma 2 to the Vecchia conditioning sets with $m_i \leq m$, the inverse computation in each factor costs at most $\mathcal{O}(m^6)$ via Cholesky factorization; importantly, this cost is completely independent of d.

Proposition 2 (Score unbiasedness)**.** *Assume the data are generated from the derivative GP model at parameter $\boldsymbol{\theta}_0$. Suppose the ordering and conditioning sets are fixed, and differentiation may be interchanged with integration. Then the reduced conditional Vecchia score satisfies*

$$\mathbb{E}_{\boldsymbol{\theta}_0} \left[\nabla_{\boldsymbol{\theta}} \tilde{\ell}_n(\boldsymbol{\theta}_0) \right] = \mathbf{0}.$$

Proposition 2 shows that the score of $\tilde{\ell}_n(\boldsymbol{\theta})$ is unbiased at the data generating parameter $\boldsymbol{\theta}_0$ for fixed ordering and conditioning sets, resulting in unbiased estimating equations even though the Vecchia product is not the full joint derivative GP likelihood. In particular, optimization of (6) targets the conditional distribution of function values given observed gradients, without evaluating the marginal density of $\mathbf{g}$. This aligns the learning objective with the inferential target of the paper, where gradients are used as conditioning information and prediction and downstream decisions depend on function values.

We hold the ordering and conditioning sets fixed when differentiating the objective with respect to the kernel and noise parameters. Since (6) is a sum over conditional factors, a uniformly sampled minibatch $\mathcal{B} \subset \{1, \ldots, n\}$ gives the unbiased score estimator

$$\widehat{\nabla} \tilde{\ell}_B(\boldsymbol{\theta}) = \frac{n}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right), \tag{7}$$

following factor subsampling strategies for Vecchia GP likelihoods [6, 17]. When the scaled metric used for neighbor selection is updated, we optionally recompute the maximin ordering and nearest-neighbor conditioning sets and continue optimization with these sets fixed. Such recomputations can be skipped for exponentially increasing numbers of iterations during optimization [21].

4.3 Prediction and Computational Complexity

Prediction. Prediction uses the same local reduced factor as training. For a new input $\mathbf{x}_\star$, choose a Vecchia conditioning set $c(\star)$ from the observed inputs (usually nearest neighbors) using the same scaled input geometry as in Section 3.3. Write $m_\star = |c(\star)|$ and form the target-centered difference matrix $\mathbf{D}_\star \in \mathbb{R}^{d \times m_\star}$ from $\mathbf{x}_\star$ and the ordered inputs in $c(\star)$. The reduced gradient vector for the prediction factor is

$$\mathbf{q}_{c(\star)} = (\mathbf{I}_{m_\star} \otimes \mathbf{D}_\star^\top) \mathbf{g}_{c(\star)} \in \mathbb{R}^{m_\star^2}.$$

We then evaluate

$$p_{\boldsymbol{\theta}} \left(f_\star \mid \mathbf{y}_{c(\star)}, \mathbf{q}_{c(\star)} \right)$$

with the reduced conditional blocks constructed as in Appendix A.3. The predictive variance is the latent variance of $f_\star$. If predicting a noisy future observation, σ_y^2 is added to this variance.

Batched implementation. We evaluate multiple target conditionals in parallel during both training and prediction. Minibatching in (7) selects the factors used to estimate the gradient, whereas batching here evaluates multiple conditionals in parallel. For each prediction target, let

$$\mathbf{K}_\star^R = \mathrm{Cov} \left(\begin{bmatrix} \mathbf{y}_{c(\star)} \\ \mathbf{q}_{c(\star)} \end{bmatrix} \right).$$

For a batch $\mathcal{B}$ of targets with the same conditioning set size m, these reduced covariance matrices are stacked into an array as

$$\mathrm{stack}_{\star \in \mathcal{B}} \left(\mathbf{K}_\star^R \right) \in \mathbb{R}^{|\mathcal{B}| \times (m+m^2) \times (m+m^2)}.$$

We assemble the reduced blocks in parallel and use batched Cholesky factorizations and triangular solves to evaluate the conditional means and variances.

Computational complexity. For a factor with at most m conditioning inputs, direct construction of the reduced blocks costs $\mathcal{O}(dm^2)$ time, while the dense Cholesky factorization of the reduced conditional covariance costs $\mathcal{O}(m^6)$. Since the objective in (6) is a sum over Vecchia factors, a stochastic update using a minibatch $\mathcal{B}$ costs

$$\mathcal{O}\left(|\mathcal{B}|(dm^2 + m^6)\right) \quad \text{time}, \qquad \mathcal{O}\left(|\mathcal{B}|(m^2 d + m^4)\right) \quad \text{memory}$$

under batched evaluation. For fixed m and fixed minibatch size, each stochastic likelihood update is linear in the input dimension d and independent of n once the ordering and conditioning sets are fixed. A full likelihood evaluation, or one complete pass over all training factors, costs $\mathcal{O}\left(n(dm^2 + m^6)\right)$.

5 Experiments

In this section, we evaluate whether TERA preserves the predictive contribution of observed gradients while reducing the cost of derivative conditioning. We first use GP simulations, where exact derivative GP posterior marginals are available, to separate posterior fidelity from model misspecification. We then evaluate large-scale GP regression, where exact derivative GP inference is no longer practical, and high-dimensional BO, where gradients inform the surrogate but decisions depend only on function values. The experiments are implemented in a PyTorch [24] code base using GPyTorch [13] and BoTorch [3] where applicable. Wall-clock time and peak GPU memory are measured on a single NVIDIA H100 GPU. Experimental details are provided in Appendix C.

5.1 Posterior Fidelity and Scaling

We first evaluate posterior fidelity in a controlled GP-simulated setting. Training and evaluation inputs are generated independently, and noiseless function values and gradients are sampled from an exact derivative GP with a Matérn-5/2 kernel. We fix the output scale to 1.0 and choose the isotropic lengthscale so that two training inputs at the median pairwise distance have prior correlation 0.3. This keeps the typical prior correlation comparable across dimensions, so that changes in posterior fidelity are not confounded by changes in effective prior smoothness as d varies. Appendix E.1 shows that the posterior-fidelity pattern is robust to the choice of target prior correlation. The exact derivative GP posterior is computed by dense algebra when feasible and by the de Roos decomposition [10] otherwise. We measure posterior fidelity by the average of marginal KL divergence $\mathrm{KL}(p_j^\star \,\|\, p_j^{\mathcal{M}})$, where $p_j^\star$ is the exact derivative GP marginal and $p_j^{\mathcal{M}}$ is the marginal from method $\mathcal{M}$.

We compare TERA with function-only Exact GP, dense Exact derivative GP (Exact dGP), Exact dGP-deRoos, full-gradient Vecchia dGP, and Vecchia dGP-deRoos. Exact dGP-deRoos computes the same posterior as dense Exact dGP using the de Roos decomposition, while Vecchia dGP-deRoos is the full-gradient Vecchia counterpart accelerated by the same algebra. The Vecchia baselines use the same ordering and conditioning sets as TERA but condition on full local gradients.

TERA preserves the gradient information that closes the gap to Exact dGP. The left column of Figure 2 shows posterior approximation error relative to the exact derivative GP posterior. Exact GP remains far from this reference, indicating that the observed gradients contribute substantial information to function-value prediction. Full-gradient Vecchia dGPs narrow the gap, with fidelity improving as m increases. Across all sweeps, TERA overlaps with the full-gradient Vecchia baselines, consistent with Proposition 1. Thus, the remaining KL gap to Exact dGP is due to Vecchia approximations, not to replacing full local gradients by reduced gradients.

TERA achieves Vecchia dGP fidelity at substantially lower cost. The middle and right columns show wall-clock time and peak GPU memory. TERA matches the posterior fidelity of full-gradient Vecchia dGPs while running much faster in the m sweep and remaining close to Exact GP timing across the d and n sweeps. Its peak memory is comparable to Exact GP for $m < 50$, while Vecchia dGP explicitly carries all md local gradient coordinates. Exact dGP becomes infeasible as d or n grows, and Exact dGP-deRoos shows rapidly increasing memory cost with n.

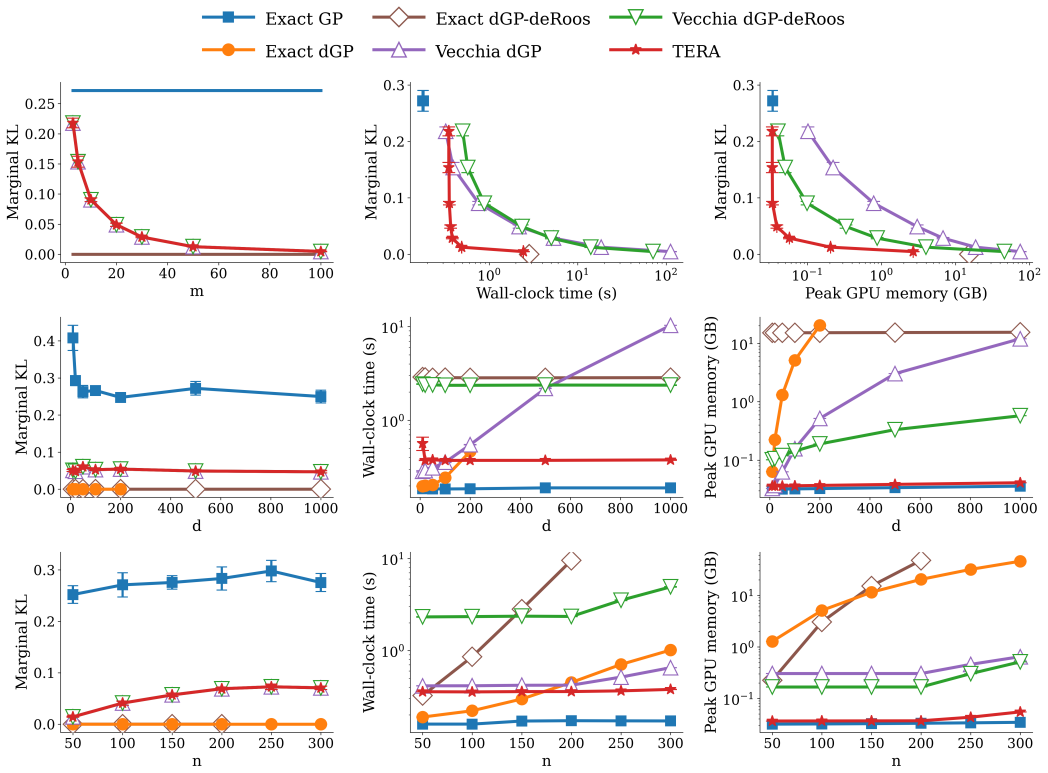

Figure 2: Posterior fidelity and computational scaling under GP simulations. Rows show sweeps over the conditioning size m at $(d, n) = (500, 150)$, input dimension d at $(m, n) = (20, 150)$, and training size n at $d = 150$ with $m = \max(20, n/10)$. Columns report average marginal KL divergence to the exact derivative GP posterior, wall-clock time, and peak GPU memory. All simulations use 100 common evaluation points and five different runs.

5.2 Large-Scale GP Regression with Observed Gradients

We evaluate large-scale GP regression on the MD22 molecular benchmark [8], following the experimental setup of [14]. Inputs are molecular configurations, responses are scalar energies, and observed forces provide gradient information. We use six benchmarks, DHA, AT-AT, Stachyose, AT-AT-CG-CG, Buckyball-catcher, and Double-walled-nanotube, spanning 5,032 to 69,753 samples and 168 to 1,110 input dimensions. The prediction target is held-out scalar energy, while gradients are used only as conditioning information.

We compare TERA with Exact GP, DDSVGP [23], and DSoftKI [14] on identical train and test splits over three different split seeds. Exact GP provides a value-only baseline, while DDSVGP and DSoftKI represent scalable derivative GP approaches. The full-gradient methods in Section 5.1 serve as fidelity controls on smaller problems. Here we focus on methods that can be applied at the scale of MD22. For DDSVGP and DSoftKI, energies and forces are standardized using the joint energy-force scale from [14] while TERA standardizes the energy target using the energy scale. We therefore report raw energy RMSE per atom after converting predictions back to physical units. Since the DDSVGP implementation does not support ARD or Matérn kernels, the main comparison uses squared exponential (SE) kernels without ARD for all methods. Additional results, including Matérn-5/2 kernels and ARD, are provided in Appendix D.

TERA achieves the strongest prediction accuracy with a lightweight learning problem. Figure 3 shows that TERA gives the lowest raw energy RMSE per atom across the MD22 systems. DDSVGP and DSoftKI use gradients through larger approximate full-derivative GP objectives, whereas TERA learns only the kernel and noise parameters in the conditional Vecchia objective. Empirically, one pass over the training factors is sufficient for TERA to obtain the reported accuracy, while DDSVGP and DSoftKI need to be trained for multiple epochs. See Figure 5 in Appendix D.

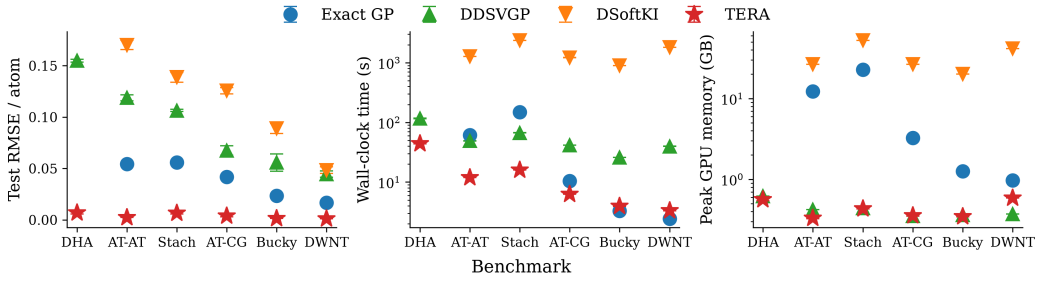

Figure 3: Large-scale scalar energy prediction on MD22. Panels show raw energy test RMSE per atom after conversion to physical units, end-to-end wall-clock time, and peak GPU memory. All methods use isotropic squared exponential kernels and identical train and test splits. Results are averaged over three independent runs. Lower values are better in all panels.

TERA remains computationally practical. The timing and memory panels show that TERA's wall-clock time and peak GPU memory usage remain among the lowest across the compared methods in both large-sample and high-dimensional regimes. On DHA, the largest system with 69,753 samples and 168 dimensions, TERA scales to the full training set while using gradients, whereas Exact GP and DSoftKI do not complete under the same resource budget. On Double-walled-nanotube, the highest dimensional system with 5,032 samples and 1,110 dimensions, TERA remains memory-efficient despite the large gradient dimension. The effect of the conditioning set size is examined in Table 5 in Appendix E.2.

5.3 High-Dimensional Bayesian Optimization

We provide BO experiments as a downstream test of whether TERA's surrogate-level gradient reduction translates into better high-dimensional scalar objective optimization with a limited number of queried locations. We consider a setting in which each query returns a scalar objective value and its full gradient. Although obtaining gradients incurs additional, application-dependent cost, we target settings in which automatic differentiation or an adjoint solver provides the full gradient without requiring d separate objective evaluations. We use two full-dimensional synthetic benchmarks, Ackley-200D and Levy-200D, covering rugged and multimodal objectives that depend on all input dimensions rather than a small embedded active subspace. Results are averaged over five independent runs with different initial designs. We report simple regret against the number of queried locations, together with end-to-end BO wall-clock time and peak GPU memory.

We compare with Sobol, dimension-scaled vanilla BO (VBO) [15], and TuRBO-1 [12]. Sobol provides a nonadaptive space-filling baseline. VBO is a modern global GP baseline for high-dimensional BO that uses lengthscale priors scaled with the input dimension, while TuRBO-1 is a strong local trust-region baseline. We use TuRBO-1 rather than multi-region variants to keep a sequential evaluation with one new query per BO iteration, without introducing allocation across multiple trust regions. All adaptive BO methods use LogEI [1], with TuRBO-1 optimizing it inside the current trust region, so the acquisition rule is aligned across GP surrogates. We also include L-BFGS-B [5] as a gradient-based local optimizer, initialized at the best point in the common Sobol initial design. It is allowed up to 300 queried locations but may terminate earlier.

TERA lowers regret at practical BO cost. Figure 4 shows that TERA obtains the lowest final regret on both high-dimensional benchmarks, converting available gradient information into substantially better BO decisions. On Levy-200D, the gap is large. VBO and TuRBO-1 improve over Sobol after the initial design, but their progress slows at substantially higher regret, while TERA continues to make larger improvements before stabilizing at a lower level. On Ackley-200D, the same ordering appears even more clearly. The value-only BO baselines reduce regret, but they plateau well above TERA, indicating that their scalar observations alone do not provide enough local information to guide the search as effectively in this 200-dimensional space. L-BFGS-B improves rapidly after initialization but terminates at higher regret on both benchmarks, showing that access to gradients alone does not ensure comparable optimization performance. The resource panel shows that TERA incorporates full gradient observations at a computational cost comparable to value-only BO. For

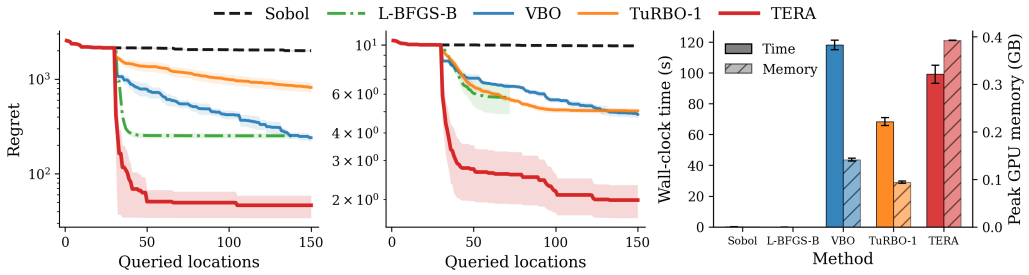

Figure 4: High-dimensional BO under a fixed budget of 150 objective evaluations. The first two panels report simple regret on full-dimensional Levy-200D and Ackley-200D, respectively. Curves show means and standard errors over five runs. The right panel reports wall-clock time and peak GPU memory on Ackley-200D, with error bars showing standard errors.

example, the full Ackley-200D run still takes only about 100 seconds (i.e., less than 1 second on average per queried location), and remains below 0.4 GB of peak GPU memory.

Comparisons with the original large-budget TuRBO setup and VBO extended to 300 queried locations are provided in Figures 8 and 9, respectively, in Appendix D.2. The effect of the conditioning set size is examined in Figure 11 in Appendix E.2.

5.4 Discussion and Limitations

TERA reduces the dependence on the input dimension, but the reduced local solve still depends on the Vecchia conditioning size. With m conditioning inputs, the reduced conditional covariance has size at most $m^2 \times m^2$, so each Vecchia factor includes an $\mathcal{O}(m^6)$ dense Cholesky term. The method can therefore be computationally costly when accurate prediction requires large conditioning sets. We use $m = 20$ as a common default in the main GP regression and BO experiments rather than tuning it for each task. An appropriate choice of m depends on the task and available computational resources. Additional ablations on the effect of m are discussed in Appendix E.2.

For general anisotropic or correlated gradient noise, conditioning on the reduced gradients still defines a valid Gaussian conditional factor, but it need not match conditioning on the full noisy gradient vector. Our experiments focus on noise settings covered by Proposition 1. The main GP regression experiments use iid unscaled gradient noise with an isotropic SE kernel, and the ARD Matérn-5/2 experiments in Appendix D.1 use metric-matched gradient noise. Exact reductions for arbitrary gradient noise remain outside the scope of this work.

6 Conclusions

We introduced TERA, a scalable derivative GP method that uses observed gradients as conditioning information for function-value prediction, rather than treating the full derivative state as the main computational object. For stationary kernels under the stated noise assumptions, each local function-value conditional depends on the full gradients only through target-specific reduced gradients, which can be embedded in Vecchia factors without changing the corresponding local conditional distribution. Empirically, the reduction matches the posterior fidelity of full-gradient Vecchia derivative GP inference and turns observed gradients into improved function prediction and lower high-dimensional BO regret at practical computational cost. The results suggest that scalable derivative GP inference should be organized around the predictive target, not around all available derivative coordinates.

Acknowledgments

This research was supported by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin–Madison with funding from the Wisconsin Alumni Research Foundation, including via a Research Forward award (PI: Philip A. Townsend).

References

[1] Sebastian Ament, Sam Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. In *Advances in Neural Information Processing Systems*, 2023.

[2] Sebastian Ament and Carla P. Gomes. Scalable first-order bayesian optimization via structured automatic differentiation. In *International Conference on Machine Learning*, 2022.

[3] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Sam Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable bayesian optimization in pytorch. In *Advances in Neural Information Processing Systems*, 2020.

[4] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18:153:1–153:43, 2015.

[5] Richard H. Byrd, Pei-Huang Lu, Jorge Nocedal, and Ci-You Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16:1190–1208, 1995.

[6] Jian Cao, Joseph Guinness, Marc G. Genton, and Matthias Katzfuss. Scalable gaussian-process regression and variable selection using vecchia approximations. *Journal of Machine Learning Research*, 23:348:1–348:30, 2022.

[7] Jian Cao, Myeongjong Kang, Felix Jimenez, Huiyan Sang, Florian Schäfer, and Matthias Katzfuss. Variational sparse inverse cholesky approximation for latent gaussian processes via double kullback-leibler minimization. In *International Conference on Machine Learning*, 2023.

[8] Stefan Chmiela, Valentin Vassilev-Galindo, Oliver T. Unke, Adil Kabylda, Huziel E. Sauceda, Alexandre Tkatchenko, and Klaus-Robert Müller. Accurate global machine learning force fields for molecules with hundreds of atoms. *Science Advances*, 9, 2022.

[9] Abhirup Datta, Sudipto Banerjee, Andrew O. Finley, and Alan E. Gelfand. Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.

[10] Filip de Roos, Alexandra Gessner, and Philipp Hennig. High-dimensional gaussian process inference with derivatives. In *International Conference on Machine Learning*, 2021.

[11] David Eriksson, Kun Dong, Eric Hans Lee, David S. Bindel, and Andrew Gordon Wilson. Scaling gaussian process regression with derivatives. In *Advances in Neural Information Processing Systems*, 2018.

[12] David Eriksson, Michael Pearce, Jacob R. Gardner, Ryan D. Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, 2019.

[13] Jacob R. Gardner, Geoff Pleiss, David S. Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[14] Daniel Huang. Scaling gaussian process regression with full derivative observations. *Transactions on Machine Learning Research*, 2026.

[15] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions. In *International Conference on Machine Learning*, 2024.

[16] Antony Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988.

[17] Felix Jimenez and Matthias Katzfuss. Scalable bayesian optimization using vecchia approximations of gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, 2022.

[18] Myeongjong Kang and Matthias Katzfuss. Correlation-based sparse inverse Cholesky factorization for fast Gaussian-process inference. *Statistics and Computing*, 33(56):1–17, 2023.

[19] Matthias Katzfuss and Joseph Guinness. A general framework for vecchia approximations of gaussian processes. *Statistical Science*, 2021.

[20] Matthias Katzfuss, Joseph Guinness, Wenlong Gong, and Daniel Zilber. Vecchia approximations of gaussian-process predictions. *Journal of Agricultural, Biological and Environmental Statistics*, 25:383 – 414, 2020.

[21] Matthias Katzfuss, Joseph Guinness, and Earl Christopher Lawrence. Scaled vecchia approximation for fast computer-model emulation. *SIAM/ASA Journal on Uncertainty Quantification*, 10:537–554, 2020.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[23] Misha Padidar, Xinran Zhu, Leo Huang, Jacob R. Gardner, and David S. Bindel. Scaling gaussian processes with derivative information using variational inference. In *Advances in Neural Information Processing Systems*, 2021.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[25] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[26] Florian Schäfer, Matthias Katzfuss, and Houman Owhadi. Sparse Cholesky factorization by Kullback-Leibler minimization. *SIAM Journal on Scientific Computing*, 43(3):A2019–A2046, 2021.

[27] Jasper Snoek, Kevin Swersky, Richard S. Zemel, and Ryan P. Adams. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014.

[28] Ilya M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Ussr Computational Mathematics and Mathematical Physics*, 7:86–112, 1967.

[29] Ercan Solak, Roderick Murray-Smith, William E. Leithead, Douglas J. Leith, and Carl Edward Rasmussen. Derivative observations in gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems*, 2002.

[30] Michael L. Stein, Zhiyi Chi, and L.J. Welty. Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society: Series B*, 66(2):275–296, 2004.

[31] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

[32] Aldo V. Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the royal statistical society series b-methodological*, 50:297–312, 1988.

[33] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric Poe Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[34] Jian Wu, Matthias Poloczek, Andrew Gordon Wilson, and P. Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems*, 2017.

Appendix

A Proofs for Section 4

A.1 Proof of Lemma 1

Proof. Condition on $\mathbf{y}_c$. Under the joint Gaussian model, $(y_i, \mathbf{q}_c)$ remains jointly Gaussian with conditional mean

$$\begin{bmatrix} \mu_{y_i|\mathbf{y}_c} \\ \boldsymbol{\mu}_{\mathbf{q}_c|\mathbf{y}_c} \end{bmatrix}$$

and conditional covariance

$$\begin{bmatrix} \sigma^2_{y_i|\mathbf{y}_c} & \mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c} \\ \mathbf{k}_{\mathbf{q}_c y_i|\mathbf{y}_c} & \mathbf{K}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c} \end{bmatrix}.$$

Applying the Gaussian conditioning formula to y_i given $\mathbf{q}_c$ under this conditional Gaussian distribution gives

$$p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c) = \mathcal{N}\left(\mu_{y_i|\mathbf{y}_c} + \mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c}\mathbf{K}^{-1}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}(\mathbf{q}_c - \boldsymbol{\mu}_{\mathbf{q}_c|\mathbf{y}_c}),\ \sigma^2_{y_i|\mathbf{y}_c} - \mathbf{k}_{y_i\mathbf{q}_c|\mathbf{y}_c}\mathbf{K}^{-1}_{\mathbf{q}_c\mathbf{q}_c|\mathbf{y}_c}\mathbf{k}_{\mathbf{q}_c y_i|\mathbf{y}_c}\right).$$

This is exactly (3). □

A.2 Proof of Proposition 1

Proof. Fix a target input $\mathbf{x}_i$ and an ordered reference set $c = \{j_1, \ldots, j_{m_c}\}$. Work in scaled input coordinates

$$\tilde{\mathbf{x}} = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{1/2}\mathbf{x}.$$

For gradients, write

$$\tilde{\nabla} f(\tilde{\mathbf{x}}) = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1/2}\nabla_{\mathbf{x}} f(\mathbf{x}), \qquad \tilde{\mathbf{g}}_j = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1/2}\mathbf{g}_j.$$

Under exact gradient observations, $\tilde{\mathbf{g}}_j = \tilde{\nabla} f(\tilde{\mathbf{x}}_j)$. Let

$$\mathcal{S}_i = \operatorname{span}\{\tilde{\mathbf{x}}_{j_a} - \tilde{\mathbf{x}}_i : a = 1, \ldots, m_c\} = \operatorname{col}(\boldsymbol{\Delta}_i).$$

Every difference between two inputs in $\{i\} \cup c$ lies in $\mathcal{S}_i$, since

$$\tilde{\mathbf{x}}_{j_a} - \tilde{\mathbf{x}}_{j_b} = (\tilde{\mathbf{x}}_{j_a} - \tilde{\mathbf{x}}_i) - (\tilde{\mathbf{x}}_{j_b} - \tilde{\mathbf{x}}_i).$$

Let $\mathcal{S}_i^{\perp}$ denote the orthogonal complement of $\mathcal{S}_i$ in scaled coordinates. For the stationary kernel

$$k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = \kappa_{\boldsymbol{\theta}}\left(\|\tilde{\mathbf{x}}_a - \tilde{\mathbf{x}}_b\|^2\right),$$

write $\mathbf{r}_{ab} = \tilde{\mathbf{x}}_a - \tilde{\mathbf{x}}_b$ and $r_{ab} = \|\mathbf{r}_{ab}\|^2$. The required derivative identities are

$$\nabla_{\tilde{\mathbf{x}}_a} k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = 2\kappa'_{\boldsymbol{\theta}}(r_{ab})\mathbf{r}_{ab}, \qquad \nabla_{\tilde{\mathbf{x}}_b} k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = -2\kappa'_{\boldsymbol{\theta}}(r_{ab})\mathbf{r}_{ab},$$

and

$$\nabla_{\tilde{\mathbf{x}}_a}\nabla^{\top}_{\tilde{\mathbf{x}}_b} k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = -2\kappa'_{\boldsymbol{\theta}}(r_{ab})\mathbf{I}_d - 4\kappa''_{\boldsymbol{\theta}}(r_{ab})\mathbf{r}_{ab}\mathbf{r}^{\top}_{ab}. \tag{8}$$

Take any $a, b \in \{i\} \cup c$ and any direction $\mathbf{u} \in \mathcal{S}_i^{\perp}$. Since $\mathbf{r}_{ab} \in \mathcal{S}_i$,

$$\mathbf{Cov}_{\boldsymbol{\theta}}\left(\mathbf{u}^{\top}\tilde{\nabla} f(\tilde{\mathbf{x}}_a), f(\mathbf{x}_b)\right) = \mathbf{u}^{\top}\nabla_{\tilde{\mathbf{x}}_a} k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = 0.$$

The same equality holds with $f(\mathbf{x}_b)$ replaced by y_b, because the function observation noise is independent of the Gaussian process and of the gradients.

Next take any $\mathbf{v} \in \mathcal{S}_i$. Using (8),

$$\mathbf{Cov}_{\boldsymbol{\theta}}\left(\mathbf{u}^{\top}\tilde{\nabla} f(\tilde{\mathbf{x}}_a), \mathbf{v}^{\top}\tilde{\nabla} f(\tilde{\mathbf{x}}_b)\right) = \mathbf{u}^{\top}\left[-2\kappa'_{\boldsymbol{\theta}}(r_{ab})\mathbf{I}_d - 4\kappa''_{\boldsymbol{\theta}}(r_{ab})\mathbf{r}_{ab}\mathbf{r}^{\top}_{ab}\right]\mathbf{v} = 0.$$

The identity term vanishes because $\mathbf{u} \perp \mathbf{v}$. The rank-one term vanishes because $\mathbf{u} \perp \mathbf{r}_{ab}$.

Decompose each scaled reference gradient into its components in $\mathcal{S}_i$ and $\mathcal{S}_i^{\perp}$. Let $\tilde{\mathbf{g}}_{c,\parallel}$ collect the components in $\mathcal{S}_i$, and let $\tilde{\mathbf{g}}_{c,\perp}$ collect the components in $\mathcal{S}_i^{\perp}$. The covariance calculations above imply

$$\mathbf{Cov}_{\boldsymbol{\theta}}\left(\tilde{\mathbf{g}}_{c,\perp}, (y_i, \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel})\right) = \mathbf{0}.$$

Since all variables are jointly Gaussian,

$$p_{\boldsymbol{\theta}}\left(y_i \mid \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel}, \tilde{\mathbf{g}}_{c,\perp}\right) = p_{\boldsymbol{\theta}}\left(y_i \mid \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel}\right).$$

It remains to show that $\mathbf{q}_c$ determines $\tilde{\mathbf{g}}_{c,\parallel}$. For any scaled reference gradient $\tilde{\mathbf{g}}_j$,

$$\boldsymbol{\Delta}_i^\top \tilde{\mathbf{g}}_j = \mathbf{D}_i^\top \mathbf{g}_j.$$

The map $\mathbf{v} \mapsto \boldsymbol{\Delta}_i^\top \mathbf{v}$ is one-to-one on $\mathcal{S}_i$. Indeed, if $\mathbf{v} \in \mathcal{S}_i$ and $\boldsymbol{\Delta}_i^\top \mathbf{v} = \mathbf{0}$, then $\mathbf{v} = \boldsymbol{\Delta}_i \mathbf{a}$ for some $\mathbf{a}$ and

$$\|\mathbf{v}\|^2 = \mathbf{a}^\top \boldsymbol{\Delta}_i^\top \boldsymbol{\Delta}_i \mathbf{a} = \mathbf{a}^\top \boldsymbol{\Delta}_i^\top \mathbf{v} = 0.$$

Thus $\mathbf{v} = \mathbf{0}$. Therefore

$$\mathbf{q}_c = (\mathbf{I}_{m_c} \otimes \mathbf{D}_i^\top)\mathbf{g}_c = (\mathbf{I}_{m_c} \otimes \boldsymbol{\Delta}_i^\top)\tilde{\mathbf{g}}_c$$

determines $\tilde{\mathbf{g}}_{c,\parallel}$, and conversely $\mathbf{q}_c$ is a deterministic function of $\tilde{\mathbf{g}}_{c,\parallel}$. Thus conditioning on $\mathbf{q}_c$ is equivalent to conditioning on $\tilde{\mathbf{g}}_{c,\parallel}$. Combining this equivalence with the conditional irrelevance of $\tilde{\mathbf{g}}_{c,\perp}$ gives

$$\begin{aligned} p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{g}_c) &= p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel}, \tilde{\mathbf{g}}_{c,\perp}) \\ &= p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel}) \\ &= p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c). \end{aligned}$$

It remains to verify that the matched noise model preserves the same decomposition. Let $\mathcal{S}_i = \mathrm{col}(\boldsymbol{\Delta}_i)$ in scaled input coordinates and write $\tilde{\mathbf{g}}_c = \tilde{\mathbf{g}}_{c,\parallel} + \tilde{\mathbf{g}}_{c,\perp}$ for the decomposition of the observed scaled gradients into components in $\mathcal{S}_i$ and $\mathcal{S}_i^\perp$. For exact gradients, the proof above shows that $\tilde{\mathbf{g}}_{c,\perp}$ is uncorrelated with $(y_i, \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel})$.

Now suppose the gradient noise is scalar in scaled coordinates,

$$\tilde{\boldsymbol{\epsilon}}_j^g = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1/2} \boldsymbol{\epsilon}_j^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \mathbf{I}_d),$$

independently across inputs. For any $\mathbf{u} \in \mathcal{S}_i^\perp$ and $\mathbf{v} \in \mathcal{S}_i$,

$$\mathbf{Cov}\left(\mathbf{u}^\top \tilde{\boldsymbol{\epsilon}}_a^g, \mathbf{v}^\top \tilde{\boldsymbol{\epsilon}}_b^g\right) = \mathbb{1}\{a = b\}\sigma_g^2 \mathbf{u}^\top \mathbf{v} = 0.$$

Thus the noise also preserves the decomposition into $\mathcal{S}_i$ and $\mathcal{S}_i^\perp$. Since the noise is independent of the GP and all variables are jointly Gaussian,

$$\tilde{\mathbf{g}}_{c,\perp} \quad \text{is independent of} \quad (y_i, \mathbf{y}_c, \tilde{\mathbf{g}}_{c,\parallel}).$$

The same argument as in the exact gradient case gives

$$p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{g}_c) = p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c).$$

The scaled noise condition above is equivalent to $\boldsymbol{\epsilon}_j^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \boldsymbol{\Lambda}_{\boldsymbol{\theta}})$ in the original gradient coordinates. Thus ARD metrics are covered by the exactness result when the gradient noise covariance is matched to the metric.

By contrast, under iid unscaled gradient noise,

$$\boldsymbol{\epsilon}_j^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \mathbf{I}_d), \qquad \tilde{\boldsymbol{\epsilon}}_j^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1}).$$

This scaled covariance is scalar only when $\boldsymbol{\Lambda}_{\boldsymbol{\theta}} \propto \mathbf{I}_d$, apart from the exact gradient case $\sigma_g^2 = 0$. Hence, for non-isotropic ARD metrics with iid unscaled gradient noise, the exact equality need not hold because the perpendicular noisy component can carry information about the parallel noisy component through the noise covariance. The reduced noisy gradient still defines a valid Gaussian conditional density, but it is not generally identical to conditioning on the full noisy gradient in this case. □

A.3 Proof of Lemma 2: Explicit Construction of Reduced Conditional Blocks

Proof. We provide the explicit algebraic formulas for the reduced blocks required in Lemma 2. Fix a target input $\mathbf{x}_i$ and a reference set $c = \{j_1, \dots, j_{m_c}\}$. Let $\mathbf{H}_i = \mathbf{D}_i^\top \mathbf{\Lambda}_{\boldsymbol{\theta}} \mathbf{D}_i \in \mathbb{R}^{m_c \times m_c}$ be the target-centered Gram matrix from Section 3.2, and let $\mathbf{e}_a$ denote the ath Euclidean basis vector in $\mathbb{R}^{m_c}$. Write, for $a, b = 1, \dots, m_c$,

$$\begin{aligned}
\mathbf{h}_a &= \mathbf{H}_i \mathbf{e}_a = \mathbf{\Delta}_i^\top (\tilde{\mathbf{x}}_{j_a} - \tilde{\mathbf{x}}_i) \in \mathbb{R}^{m_c}, \\
r_a &= (\mathbf{x}_i - \mathbf{x}_{j_a})^\top \mathbf{\Lambda}_{\boldsymbol{\theta}} (\mathbf{x}_i - \mathbf{x}_{j_a}) = [\mathbf{H}_i]_{aa}, \\
r_{ab} &= (\mathbf{x}_{j_a} - \mathbf{x}_{j_b})^\top \mathbf{\Lambda}_{\boldsymbol{\theta}} (\mathbf{x}_{j_a} - \mathbf{x}_{j_b}) = [\mathbf{H}_i]_{aa} + [\mathbf{H}_i]_{bb} - 2[\mathbf{H}_i]_{ab}.
\end{aligned}$$

Here r_a and r_{ab} are the stationary kernel inputs for target-reference and reference-reference pairs, respectively. Let

$$\mathbf{q}_a = \mathbf{D}_i^\top \mathbf{g}_{j_a} \in \mathbb{R}^{m_c}, \qquad a = 1, \dots, m_c,$$

so that $\mathbf{q}_c = (\mathbf{q}_1^\top, \dots, \mathbf{q}_{m_c}^\top)^\top$. The required unconditioned reduced blocks are obtained directly as follows. For a reference function value y_{j_b},

$$\mathbf{k}_{\mathbf{q}_a y_{j_b}} = 2\kappa'_{\boldsymbol{\theta}}(r_{ab})(\mathbf{h}_a - \mathbf{h}_b).$$

For the target function value y_i,

$$\mathbf{k}_{\mathbf{q}_a y_i} = 2\kappa'_{\boldsymbol{\theta}}(r_a)\mathbf{h}_a.$$

For the covariance between the projected gradients at reference inputs j_a and j_b,

$$\mathbf{K}_{\mathbf{q}_a \mathbf{q}_b} = -2\kappa'_{\boldsymbol{\theta}}(r_{ab})\mathbf{H}_i - 4\kappa''_{\boldsymbol{\theta}}(r_{ab})(\mathbf{h}_a - \mathbf{h}_b)(\mathbf{h}_a - \mathbf{h}_b)^\top + \mathbb{1}\{a = b\}\sigma_g^2 \mathbf{H}_i.$$

The final term corresponds to metric-matched gradient noise and vanishes for exact gradients. Stacking these blocks over $a, b = 1, \dots, m_c$ gives $\mathbf{K}_{\mathbf{q}_c \mathbf{y}_c} \in \mathbb{R}^{m_c^2 \times m_c}$, $\mathbf{k}_{\mathbf{q}_c y_i} \in \mathbb{R}^{m_c^2}$, and $\mathbf{K}_{\mathbf{q}_c \mathbf{q}_c} \in \mathbb{R}^{m_c^2 \times m_c^2}$.

The projected mean is

$$\boldsymbol{\mu}_{\mathbf{q}_c} = (\mathbf{I}_{m_c} \otimes \mathbf{D}_i^\top)\boldsymbol{\mu}_{\mathbf{g}_c}, \qquad \boldsymbol{\mu}_{\mathbf{g}_c} = \left(\nabla \mu_{\boldsymbol{\theta}}(\mathbf{x}_{j_1})^\top, \dots, \nabla \mu_{\boldsymbol{\theta}}(\mathbf{x}_{j_{m_c}})^\top\right)^\top.$$

The conditional quantities used in evaluating the conditional density are ordinary Gaussian Schur complements with respect to $\mathbf{y}_c$:

$$\begin{aligned}
\mathbf{K}_{\mathbf{q}_c \mathbf{q}_c | \mathbf{y}_c} &= \mathbf{K}_{\mathbf{q}_c \mathbf{q}_c} - \mathbf{K}_{\mathbf{q}_c \mathbf{y}_c} \mathbf{K}_{\mathbf{y}_c \mathbf{y}_c}^{-1} \mathbf{K}_{\mathbf{y}_c \mathbf{q}_c}, \\
\mathbf{k}_{\mathbf{q}_c y_i | \mathbf{y}_c} &= \mathbf{k}_{\mathbf{q}_c y_i} - \mathbf{K}_{\mathbf{q}_c \mathbf{y}_c} \mathbf{K}_{\mathbf{y}_c \mathbf{y}_c}^{-1} \mathbf{k}_{\mathbf{y}_c y_i},
\end{aligned}$$

and

$$\boldsymbol{\mu}_{\mathbf{q}_c | \mathbf{y}_c} = \boldsymbol{\mu}_{\mathbf{q}_c} + \mathbf{K}_{\mathbf{q}_c \mathbf{y}_c} \mathbf{K}_{\mathbf{y}_c \mathbf{y}_c}^{-1} (\mathbf{y}_c - \boldsymbol{\mu}_{\mathbf{y}_c}).$$

Here $\mathbf{K}_{\mathbf{y}_c \mathbf{y}_c} \in \mathbb{R}^{m_c \times m_c}$ includes the function observation noise. These formulas provide all matrix and vector quantities needed for the corrections without materializing the $m_c d \times m_c d$ local derivative covariance. □

A.4 Proof of Proposition 2

Proof. Let

$$\ell_i(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}\left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)}\right), \qquad \tilde{\ell}_n(\boldsymbol{\theta}) = \sum_{i=1}^n \ell_i(\boldsymbol{\theta}).$$

The ordering and conditioning sets are fixed. Therefore $\mathbf{q}_{c(i)}$ is a deterministic function of the observed gradients and inputs for each i.

It is enough to show that each local score has mean zero at $\boldsymbol{\theta}_0$. By the tower property,

$$\mathop{\mathbb{E}}_{\boldsymbol{\theta}_0}\left[\nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}_0)\right] = \mathop{\mathbb{E}}_{\boldsymbol{\theta}_0}\left[\mathop{\mathbb{E}}_{\boldsymbol{\theta}_0}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}\left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)}\right)\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)}\right]\right].$$

Conditioning on $(\mathbf{y}_{c(i)}, \mathbf{q}_{c(i)})$, the inner expectation is

$$\int \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}\left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)}\right)\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} p_{\boldsymbol{\theta}_0}\left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)}\right) dy_i.$$

Using

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right) = \frac{\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right)}{p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right)},$$

the inner expectation equals

$$\int \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} dy_i.$$

By the assumed interchange of differentiation and integration,

$$\int \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} dy_i = \nabla_{\boldsymbol{\theta}} \int p_{\boldsymbol{\theta}} \left(y_i \mid \mathbf{y}_{c(i)}, \mathbf{q}_{c(i)} \right) \, dy_i \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}.$$

The integral is one for every $\boldsymbol{\theta}$, since the integrand is a conditional density in y_i. Therefore the last term is zero, and hence

$$\mathop{\mathbb{E}}_{\boldsymbol{\theta}_0} \left[\nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}_0) \right] = \mathbf{0}.$$

Summing over i gives

$$\mathop{\mathbb{E}}_{\boldsymbol{\theta}_0} \left[\nabla_{\boldsymbol{\theta}} \tilde{\ell}_n(\boldsymbol{\theta}_0) \right] = \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\boldsymbol{\theta}_0} \left[\nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}_0) \right] = \mathbf{0}.$$

This proves Proposition 2. □

B Extensions Beyond Stationary Kernels

B.1 Dot-Product Kernels

The same span argument extends to dot-product kernels, with input spans replacing target-centered difference spans. Consider covariance kernels of the form

$$k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = \psi_{\boldsymbol{\theta}} \left(\mathbf{x}_a^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}} \mathbf{x}_b + b_{\boldsymbol{\theta}} \right),$$

where $\boldsymbol{\Lambda}_{\boldsymbol{\theta}} \succ 0$ is symmetric positive definite and $\psi_{\boldsymbol{\theta}}$ is twice continuously differentiable. Suppose that the observed gradients are either exact or have independent Gaussian noise $\boldsymbol{\epsilon}_j^g \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \boldsymbol{\Lambda}_{\boldsymbol{\theta}})$, independent of the GP and function observation noise.

Fix a target input $\mathbf{x}_i$ and reference set $c = \{j_1, \ldots, j_{m_c}\}$ with $i \notin c$. In scaled coordinates $\tilde{\mathbf{x}} = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{1/2} \mathbf{x}$, define

$$\tilde{\mathbf{X}}_i = [\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_{j_1}, \ldots, \tilde{\mathbf{x}}_{j_{m_c}}] \in \mathbb{R}^{d \times (m_c+1)}, \qquad \mathcal{S}_i = \mathrm{col}(\tilde{\mathbf{X}}_i).$$

Writing $s_{ab} = \tilde{\mathbf{x}}_a^\top \tilde{\mathbf{x}}_b + b_{\boldsymbol{\theta}}$, the derivative identities are

$$\nabla_{\tilde{\mathbf{x}}_a} k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = \psi'_{\boldsymbol{\theta}}(s_{ab}) \tilde{\mathbf{x}}_b$$

and

$$\nabla_{\tilde{\mathbf{x}}_a} \nabla_{\tilde{\mathbf{x}}_b}^\top k_{\boldsymbol{\theta}}(\mathbf{x}_a, \mathbf{x}_b) = \psi'_{\boldsymbol{\theta}}(s_{ab}) \mathbf{I}_d + \psi''_{\boldsymbol{\theta}}(s_{ab}) \tilde{\mathbf{x}}_b \tilde{\mathbf{x}}_a^\top.$$

For $a, b \in \{i\} \cup c$, both scaled inputs lie in $\mathcal{S}_i$. Consequently, gradient components in $\mathcal{S}_i^\perp$ are uncorrelated with the target and reference function values and with gradient components in $\mathcal{S}_i$. Under the matched noise model, the scaled gradient noise has covariance $\sigma_g^2 \mathbf{I}_d$, which preserves these zero cross-covariances. Joint Gaussianity therefore makes the orthogonal gradient components independent of the function values and retained gradient components.

Let $\tilde{\mathbf{g}}_j = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1/2} \mathbf{g}_j$. A corresponding reduced gradient is

$$\mathbf{q}_c^{\mathrm{dot}} = (\mathbf{I}_{m_c} \otimes \tilde{\mathbf{X}}_i^\top) \tilde{\mathbf{g}}_c \in \mathbb{R}^{m_c(m_c+1)}.$$

The map $\mathbf{v} \mapsto \tilde{\mathbf{X}}_i^\top \mathbf{v}$ is one-to-one on $\mathcal{S}_i$, by the same argument as in Appendix A.2. Thus this statistic determines the retained gradient components, giving

$$p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{g}_c) = p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c^{\mathrm{dot}}).$$

Moreover,

$$\tilde{\mathbf{X}}_i^\top \tilde{\mathbf{g}}_j = [\mathbf{x}_i, \mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_{m_c}}]^\top \mathbf{g}_j,$$

so the statistic can be formed directly from the original inputs and observed gradients.

B.2 Input and Feature Transformations

We next consider nonstationary kernels obtained by composing a stationary base kernel with an input or feature transformation. Let $\phi : \mathcal{D} \to \mathbb{R}^p$ be a fixed differentiable transformation and write $f(\mathbf{x}) = \tilde{f}(\phi(\mathbf{x}))$, where $\tilde{f}$ is a GP with a stationary kernel of the form in (1). The resulting kernel is

$$k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = \kappa_{\boldsymbol{\theta}}\left((\phi(\mathbf{x}) - \phi(\mathbf{x}'))^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}}(\phi(\mathbf{x}) - \phi(\mathbf{x}'))\right),$$

where $\boldsymbol{\Lambda}_{\boldsymbol{\theta}} \in \mathbb{R}^{p\times p}$ is symmetric positive definite.

Write $\mathbf{u}_j = \phi(\mathbf{x}_j)$ and let $\mathbf{J}_j = \partial\phi(\mathbf{x}_j)/\partial\mathbf{x}_j^\top \in \mathbb{R}^{p\times d}$. For exact gradients, the chain rule gives

$$\mathbf{g}_j = \mathbf{J}_j^\top \nabla_{\mathbf{u}} \tilde{f}(\mathbf{u}_j).$$

For a target input $\mathbf{x}_i$ and reference set $c = \{j_1, \ldots, j_{m_c}\}$ with $i \notin c$, define

$$\mathbf{D}_i^\phi = [\mathbf{u}_{j_1} - \mathbf{u}_i, \ldots, \mathbf{u}_{j_{m_c}} - \mathbf{u}_i] \in \mathbb{R}^{p\times m_c}.$$

Proposition 1 applies in the transformed input space and retains the directional derivatives $(\mathbf{D}_i^\phi)^\top \nabla_{\mathbf{u}} \tilde{f}(\mathbf{u}_j)$ at each reference input. However, different transformed gradients can yield the same observed gradient $\mathbf{g}_j$ through the map $\mathbf{J}_j^\top$, while giving different values of $(\mathbf{D}_i^\phi)^\top \nabla_{\mathbf{u}} \tilde{f}(\mathbf{u}_j)$. To recover these projections from $\mathbf{g}_j$, we assume that, for every $j \in c$, there exists $\mathbf{A}_j \in \mathbb{R}^{d\times m_c}$ satisfying

$$\mathbf{J}_j \mathbf{A}_j = \mathbf{D}_i^\phi.$$

Then

$$\mathbf{A}_j^\top \mathbf{g}_j = (\mathbf{D}_i^\phi)^\top \nabla_{\mathbf{u}} \tilde{f}(\mathbf{u}_j),$$

and stacking these projections gives

$$\mathbf{q}_c^\phi = \begin{bmatrix} \mathbf{A}_{j_1}^\top \mathbf{g}_{j_1} \\ \vdots \\ \mathbf{A}_{j_{m_c}}^\top \mathbf{g}_{j_{m_c}} \end{bmatrix} \in \mathbb{R}^{m_c^2}.$$

By Proposition 1, the target conditional given the full transformed gradients depends on them only through $\mathbf{q}_c^\phi$. Since $\mathbf{g}_c$ is a function of those gradients and $\mathbf{q}_c^\phi$ is a function of $\mathbf{g}_c$, it follows that

$$p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{g}_c) = p_{\boldsymbol{\theta}}(y_i \mid \mathbf{y}_c, \mathbf{q}_c^\phi).$$

The recoverability condition is satisfied whenever each $\mathbf{J}_j$ has full row rank, in which case one may take

$$\mathbf{A}_j = \mathbf{J}_j^\top (\mathbf{J}_j \mathbf{J}_j^\top)^{-1} \mathbf{D}_i^\phi.$$

This includes feature transformations with $p < d$ whose Jacobians have full row rank at the reference inputs. For input warping with $p = d$ and nonsingular Jacobians, the expression simplifies to $\mathbf{A}_j = \mathbf{J}_j^{-1} \mathbf{D}_i^\phi$. Coordinatewise warping satisfies this condition when each coordinate derivative is nonzero at the reference inputs.

Under the same recoverability condition, the reduction also remains exact for noisy gradients of the form

$$\mathbf{g}_j = \mathbf{J}_j^\top \left(\nabla_{\mathbf{u}} \tilde{f}(\mathbf{u}_j) + \boldsymbol{\eta}_j\right), \qquad \boldsymbol{\eta}_j \sim \mathcal{N}(\mathbf{0}, \sigma_g^2 \boldsymbol{\Lambda}_{\boldsymbol{\theta}}),$$

where the noise vectors are independent across inputs and independent of the GP and function observation noise. Indeed,

$$\mathbf{A}_j^\top \mathbf{g}_j = (\mathbf{D}_i^\phi)^\top \left(\nabla_{\mathbf{u}} \tilde{f}(\mathbf{u}_j) + \boldsymbol{\eta}_j\right),$$

so the preceding argument applies to the noisy transformed gradients under the matched noise model of Proposition 1. The corresponding gradient noise covariance in the original coordinates is $\sigma_g^2 \mathbf{J}_j^\top \boldsymbol{\Lambda}_{\boldsymbol{\theta}} \mathbf{J}_j$. However, iid noise in those coordinates need not satisfy this condition.

C Experimental Details

C.1 Posterior Fidelity and Scaling

For the dimension sweep, a fixed lengthscale would also change the effective prior smoothness. For example, under a squared exponential (SE) covariance function, the correlation between two inputs $\mathbf{x}$ and $\mathbf{x}'$ is

$$\rho_\ell^{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right).$$

If ℓ is fixed while typical squared distances $\|\mathbf{x} - \mathbf{x}'\|^2$ grow with d, prior correlations decrease automatically. The dimension sweep would then mix posterior fidelity with a change in the prior correlation structure caused by increasing dimension. We therefore choose the lengthscale through a fixed median distance correlation criterion rather than treating it as a method tuning parameter.

Let $\{\mathbf{x}_i\}_{i=1}^n$ denote the training inputs, generated from a Sobol sequence [28], and define

$$r_{\mathrm{med}} = \mathrm{median}_{1 \le a < b \le n} \|\mathbf{x}_a - \mathbf{x}_b\|.$$

For the Matérn-5/2 covariance used in the simulations, the correlation between two inputs $\mathbf{x}$ and $\mathbf{x}'$ is

$$\rho_\ell(\mathbf{x}, \mathbf{x}') = \left(1 + \sqrt{5}\frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} + \frac{5\|\mathbf{x} - \mathbf{x}'\|^2}{3\ell^2}\right) \exp\left(-\sqrt{5}\frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right).$$

The isotropic lengthscale $\ell > 0$ is selected to satisfy

$$\left(1 + \sqrt{5}\frac{r_{\mathrm{med}}}{\ell} + \frac{5r_{\mathrm{med}}^2}{3\ell^2}\right) \exp\left(-\sqrt{5}\frac{r_{\mathrm{med}}}{\ell}\right) = \rho,$$

where $\rho = 0.3$ in the main experiments. For each run and sweep setting, the resulting ℓ is shared by all methods.

C.2 Large-Scale Regression with Observed Gradients

Let $\mathbf{R}_i$ denote the Cartesian coordinate vector of the ith molecular configuration, let E_i be its molecular energy, and let $\mathbf{F}_i = -\nabla_{\mathbf{R}} E_i$ be the force. All methods use scaled inputs $\mathbf{x}_i = \mathbf{R}_i/3$. Let $\bar{E}_{\mathrm{tr}}$ and s_E denote the training energy mean and standard deviation. For TERA and Exact GP, we standardize the energy using the training energy standard deviation. The corresponding function value and gradient are

$$y_i = -\frac{E_i - \bar{E}_{\mathrm{tr}}}{s_E}, \qquad \mathbf{g}_i = \nabla_{\mathbf{x}_i} y_i = \frac{3\mathbf{F}_i}{s_E}.$$

A predicted normalized function value $\widehat{y}_i$ is converted back to physical energy units by

$$\widehat{E}_i = \bar{E}_{\mathrm{tr}} - s_E \widehat{y}_i.$$

For DDSVGP and DSoftKI, following [14], energies and forces are normalized by a joint energy and force standard deviation s_{EF}. Their observed function value and gradient are

$$y_i^{EF} = -\frac{E_i - \bar{E}_{\mathrm{tr}}}{s_{EF}}, \qquad \mathbf{g}_i^{EF} = \nabla_{\mathbf{x}_i} y_i^{EF} = \frac{3\mathbf{F}_i}{s_{EF}}.$$

A predicted normalized function value $\widehat{y}_i^{EF}$ is converted back by

$$\widehat{E}_i = \bar{E}_{\mathrm{tr}} - s_{EF} \widehat{y}_i^{EF}.$$

We compute raw energy RMSE after converting predictions back to physical energy units and use it as the main metric for fair comparison across methods. In our experiments, we report raw energy RMSE per atom to make errors comparable across molecular systems with different numbers of atoms. We use single precision for preprocessing and model computations across all methods to allow memory-intensive baselines to run on larger datasets. See Table 1 for settings.

For TERA, each training minibatch contains 32 conditional factors, which are evaluated in parallel. Prediction likewise processes 32 targets at a time. The same batch sizes are used in the additional ARD Matérn-5/2 experiments in Appendix D.

Table 1: MD22 regression settings for the main experiments.

Common settings	
Train/test split	$90/10$
Seeds	$6535,\ 8830,\ 92357$
Input scaling	Cartesian coordinates $\mathbf{R}$ are scaled as $\mathbf{x} = \mathbf{R}/3$
Kernel	Isotropic SE
Initial lengthscale	1.0
Initial value-noise parameter	10^{-3}
Initial gradient-noise parameter	$10^{-3}d$ for DSoftKI, 10^{-3} otherwise
Hyperparameter optimization	Adam [22], weight decay 0.0
Standardization before training	
Exact GP	Training energy standard deviation s_E
TERA	Training energy standard deviation s_E
DDSVGP	Joint energy-force training standard deviation s_{EF}
DSoftKI	Joint energy-force training standard deviation s_{EF}
Method-specific optimization settings	
Exact GP	50 full-batch steps, lr $\in \{0.01, 0.005\}$
TERA	1 epoch, conditioning set size 20, batch size 32, lr 0.01
DDSVGP	50 epochs, 2 inducing directions, dimension dependent batch size and lr
DSoftKI	50 epochs, 512 interpolation points, dimension dependent batch size and lr
Dimension dependent schedule for DDSVGP and DSoftKI	
$d < 180$	batch size 1,024, DDSVGP lr 0.012, DSoftKI lr 0.008
$180 \leq d < 300$	batch size 512, DDSVGP lr 0.006, DSoftKI lr 0.004
$300 \leq d < 1000$	batch size 256, DDSVGP lr 0.003, DSoftKI lr 0.002
$d \geq 1000$	batch size 128, DDSVGP lr 0.0015, DSoftKI lr 0.001

C.3 High-Dimensional Bayesian Optimization

We evaluate high-dimensional BO on Levy-200D and Ackley-200D over $[0, 1]^d$. All methods use the same initial designs, evaluation budgets, and random seeds. Objective values are standardized before surrogate fitting. For TERA, observed gradients are divided by the same response standard deviation. All experiments are run in double precision on a single GPU. Table 2 summarizes the common setup.

Table 2: Common setup for the high-dimensional BO experiments.

Setting	Value
Benchmarks	Levy-200D, Ackley-200D
Domain	$[0, 1]^d$
Evaluation budget	150 queried locations, including the initial design
Initial design	30 Sobol points
New locations per BO iteration	1
Random seeds	$1, 27, 42, 86, 99$
Precision	double

VBO follows the implementation of [15], with an ARD SE GP and a dimension-scaled log-normal lengthscale prior. TuRBO-1 uses a local ARD Matérn-5/2 GP inside the TuRBO-1 trust region and optimizes LogEI within the current trust region. TERA uses an ARD Matérn-5/2 derivative GP surrogate with $m = 20$ conditioning inputs. The lengthscale and outputscale are initialized to 1.0. The function-value and gradient observation-noise variances are fixed at 10^{-3} for numerical stability. Table 3 gives the method-specific settings.

For TERA, kernel hyperparameters are learned by maximizing the conditional Vecchia likelihood in (6). We run 50 Adam steps after the initial Sobol design, followed by 10 warm-started Adam steps

Table 3: Surrogate fitting and acquisition optimization settings for the main experiments.

Method	Surrogate	Model fitting	Acquisition optimization
VBO	ARD SE GP	MAP estimation with a dimension-scaled lengthscale prior	LogEI, 512 raw candidates, 4 restarts, up to 300 L-BFGS iterations
TuRBO-1	Local ARD Matérn-5/2 GP	50 Adam steps per BO iteration	LogEI, 512 raw candidates, 4 restarts, up to 300 L-BFGS iterations
TERA	ARD Matérn-5/2 derivative GP	50 initial Adam steps, then 10 Adam steps every 20 BO iterations	LogEI, 256 raw candidates, 4 restarts, up to 30 L-BFGS iterations

every 20 BO iterations. The learning rate is 0.01, and each stochastic update uses up to 32 conditional factors. Training factors and prediction targets are evaluated in batches of up to 32. Between these updates, the surrogate is rebuilt with the newly observed function values and gradients while the kernel hyperparameters are held fixed. This separates the incorporation of new derivative observations from the re-estimation of the scaled local geometry. In the BO setting, frequent hyperparameter updates can repeatedly change the metric used for neighbor selection and the reduced local conditionals, which in turn changes the acquisition landscape. The periodic warm-started schedule keeps this geometry stable over short BO windows while still allowing the surrogate parameters to adapt as more data are collected. During acquisition optimization, the candidate-specific neighbor set is treated as fixed within each local predictive evaluation, making the LogEI objective piecewise differentiable.

D Additional Experimental Results

This section provides additional empirical results that complement the main experiments. We first present further GP regression results with observed gradients, followed by additional high-dimensional BO results.

D.1 Large-Scale Regression with Observed Gradients

Isotropic SE Kernel. Table 4 gives the final test errors across all six systems. TERA is the best method on every dataset, with raw RMSE per atom between 0.0012 and 0.0069. Exact GP is consistently worse than TERA on every dataset where it is run. DDSVGP and DSoftKI use derivative information, but they remain substantially less accurate than TERA, and in several cases even less accurate than Exact GP. Figure 5 shows that the test RMSE of TERA starts below the other derivative baselines and continues to decrease, while DDSVGP improves slowly and DSoftKI eventually deteriorates after an initial decrease.

Table 4: MD22 test energy prediction using an SE kernel with a single isotropic lengthscale for six benchmarks. Entries report raw energy RMSE per atom. Parentheses under each dataset give (n, d), where n is the number of molecular configurations and d is the input dimension.

	DHA $(69753, 168)$	AT-AT $(20001, 180)$	Stachyose $(27272, 261)$	AT-AT-CG-CG $(10153, 354)$	Buckyball $(6102, 444)$	DWNT $(5032, 1110)$
Exact GP	–	0.054 ± 0.001	0.056 ± 0.002	0.042 ± 0.001	0.024 ± 0.000	0.017 ± 0.001
DDSVGP	0.155 ± 0.001	0.119 ± 0.003	0.107 ± 0.001	0.068 ± 0.005	0.056 ± 0.009	0.045 ± 0.003
DSoftKI	–	0.170 ± 0.004	0.139 ± 0.005	0.126 ± 0.003	0.089 ± 0.005	0.048 ± 0.003
TERA	**0.007** ± 0.001	**0.002** ± 0.000	**0.007** ± 0.001	**0.004** ± 0.001	**0.001** ± 0.000	**0.001** ± 0.000

Buckyball denotes buckyball-catcher, and DWNT denotes double-walled-nanotube. Dashes indicate settings that were infeasible due to the resource constraint.

ARD Matérn-5/2 Kernel. As shown in Figure 6, TERA achieves the lowest test RMSE per atom on every benchmark using an ARD Matérn-$5/2$ kernel. Despite using full gradient observations, its runtime remains comparable to that of Exact GP, and its peak GPU memory stays below 1 GB across all six datasets. Figure 7 shows the corresponding training curves.

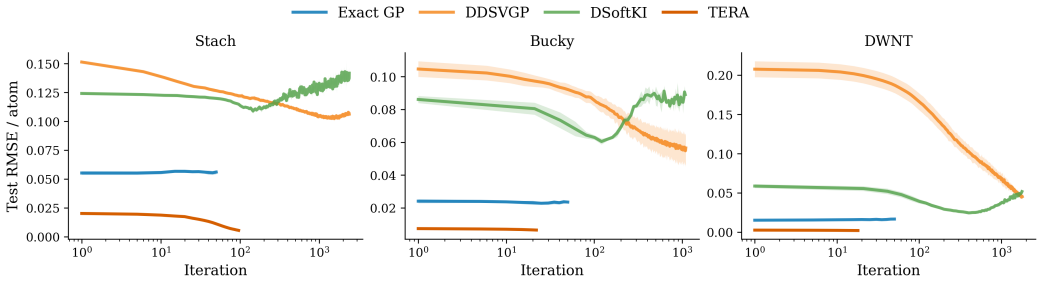

Figure 5: Training curves on selected MD22 benchmarks using an isotropic SE kernel. Each panel reports test RMSE per atom as a function of training step. TERA reaches the lowest error within its shorter training schedule.

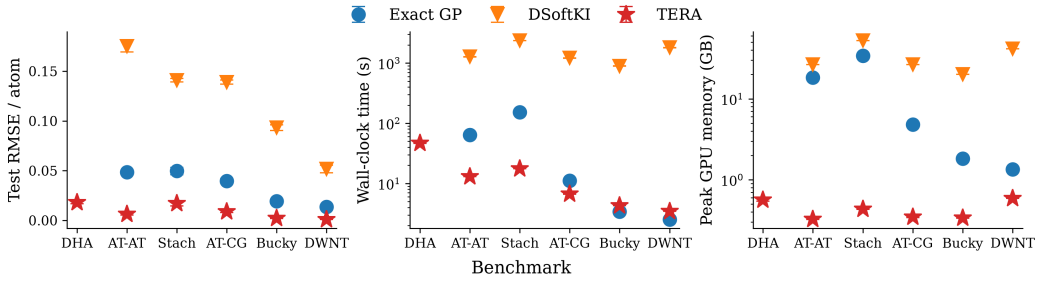

Figure 6: MD22 results with an ARD Matérn-5/2 kernel. Panels show raw energy test RMSE per atom, end-to-end wall-clock time, and peak GPU memory. TERA achieves the lowest test error on every benchmark. DDSVGP is omitted because the implementation does not support ARD Matérn kernels.

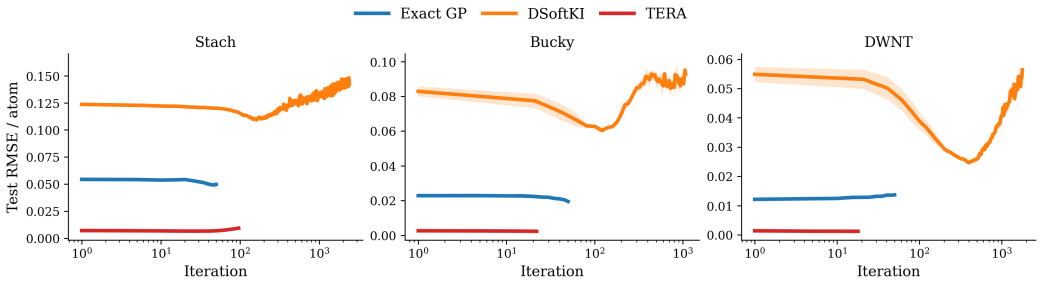

Figure 7: Training curves on selected MD22 datasets using an ARD Matérn-5/2 kernel. Each panel presents test RMSE per atom as a function of training step. TERA reaches the lowest error within its shorter training schedule.

D.2 High-Dimensional Bayesian Optimization

Comparison with the original TuRBO. The main BO comparison uses 150 queried locations. However, the original TuRBO-1 evaluation on Ackley-200D uses a much larger query budget and Thompson sampling (TS) [31] inside the trust region. We therefore include an additional comparison against TuRBO-1 TS in this original large-budget setup. TuRBO-1 TS uses 15,000 queried locations, including 200 initial points, while TERA keeps the main setting of 150 queried locations, including 30 initial points. This experiment tests whether a strong value-only local BO method can close the gap when it is allowed orders of magnitude more function evaluations.

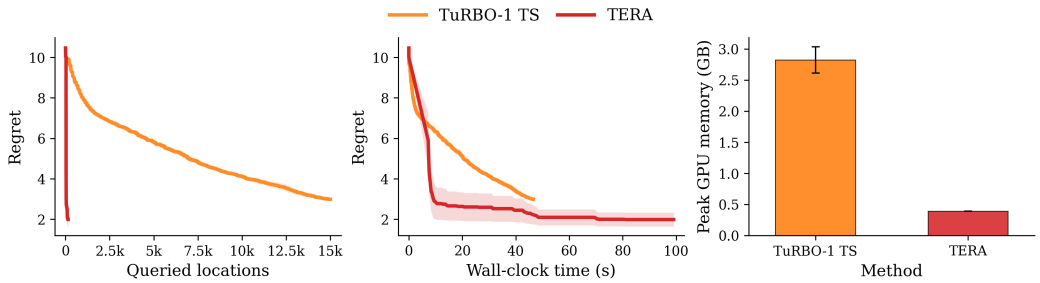

Figure 8: Additional comparison on Ackley-200D against TuRBO-1 TS in a large-budget setting. The left panel shows simple regret against the number of objective queries. The middle panel presents simple regret against end-to-end wall-clock time. The right panel reports peak GPU memory. TERA uses the same setting as in the main experiments.

Figure 8 shows that the larger query budget substantially improves TuRBO-1 TS, but it does not eliminate the advantage of using gradients through TERA. Using only one hundredth as many queried locations, TERA achieves a final mean regret of 1.99, compared with 2.99 for TuRBO-1 TS. The wall-clock comparison shows that TuRBO-1 TS completes its larger budget in less total time, but TERA already reaches lower regret before the TuRBO-1 TS run finishes. The memory panel shows that TERA uses far less peak GPU memory than TuRBO-1 TS in this experiment, approximately 0.39 GB versus 2.82 GB.

Extended VBO and gradient-based optimization. We further compare TERA with VBO extended to 300 queried locations and with L-BFGS-B initialized at the best point in the common Sobol design. This comparison examines whether additional value-only evaluations or direct use of gradients for local optimization can achieve similar performance. TERA keeps its main setting of 150 queried locations.

Figure 9 shows that TERA retains lower final regret than VBO on both benchmarks, despite using half as many queried locations. On Ackley-200D, TERA achieves mean regret 1.99 in about 99 seconds, compared with 3.72 in 318 seconds for VBO. On Levy-200D, the corresponding values are 46.49 in 130 seconds for TERA and 110.01 in 344 seconds for VBO. Thus, the advantage persists when VBO is given both more queried locations and more wall-clock time. L-BFGS-B improves rapidly after initialization and incurs negligible runtime on these inexpensive synthetic objectives, but terminates at substantially higher regret. Its final mean regrets are 5.80 on Ackley-200D and 252.06 on Levy-200D. These results show that using gradients within TERA's surrogate supports more effective global search than using them directly for local descent.

E Ablation Study

E.1 Sensitivity to Kernel Smoothness

Figure 10 shows how the posterior fidelity results vary with the correlation level used to calibrate the simulated GP prior. Across all values of ρ, Exact dGP-deRoos remains essentially at zero KL, confirming that it recovers the exact derivative GP reference. Exact GP has a substantially larger discrepancy, especially for smoother priors with larger ρ, reflecting the predictive information lost when gradient observations are ignored. In contrast, TERA remains much closer to the exact derivative

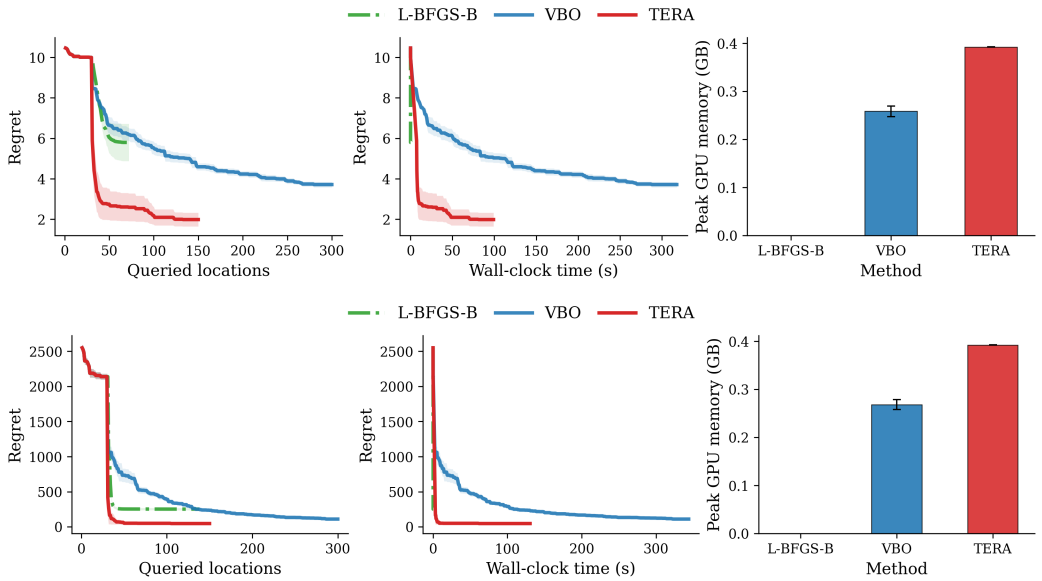

Figure 9: Additional comparisons with VBO and L-BFGS-B on Ackley-200D (top) and Levy-200D (bottom). The left and middle panels show simple regret against queried locations and end-to-end wall-clock time, respectively. The right panels report peak GPU memory. TERA uses 150 queried locations, while VBO and L-BFGS-B are allowed up to 300, including the common initial design. L-BFGS-B may terminate before exhausting its budget. Curves and bars show means over five runs, with shaded regions and error bars indicating standard errors.

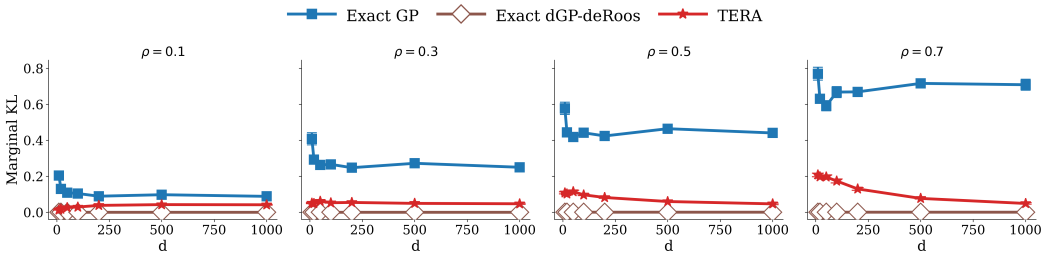

Figure 10: Sensitivity to the target median prior correlation in the GP simulation d sweep. For each panel, the Matérn-5/2 lengthscale is calibrated so that two training inputs at the median pairwise distance have prior correlation ρ. With $m = 20$ fixed, we vary $\rho \in \{0.1, 0.3, 0.5, 0.7\}$ and report average marginal KL divergence to the exact derivative GP posterior as the input dimension d increases. The setting $\rho = 0.3$ is used in the main experiments.

GP posterior across the tested dimensions. The results confirm that its posterior fidelity is stable across both rougher and smoother simulated priors.

E.2 Effect of Vecchia Conditioning Set Size

Table 5 evaluates the tradeoff induced by the Vecchia conditioning set size m across all six MD22 datasets. Increasing m gives each local conditional access to a larger neighborhood, which can improve the approximation quality of the conditional Vecchia surrogate. At the same time, the reduced local covariance blocks have size at most $m^2 \times m^2$, so the cost of the dense local solve grows quickly with m. Increasing m from 5 to 20 reduces mean test RMSE on all six datasets, although the improvement is not monotone at intermediate values. Beyond $m = 20$, accuracy gains are small or inconsistent, while runtime and peak GPU memory increase substantially. For example, on DHA, increasing m from 20 to 50 raises runtime from approximately 46 to 650 seconds and peak GPU memory from 0.54 to 5.57 GB without improving test RMSE. For small conditioning sets, peak

Table 5: Effect of the conditioning set size m on TERA for all six MD22 datasets using an isotropic Matérn-5/2 kernel. The metrics are raw energy test RMSE per atom, end-to-end wall-clock time, and peak GPU memory. Results report the mean $\pm$ standard error over three runs. Lower is better for all metrics.

Dataset	Metric	$m = 5$	$m = 10$	$m = 20$	$m = 30$	$m = 50$
DHA	RMSE/atom	0.0217 ± 0.0007	0.0218 ± 0.0006	0.0193 ± 0.0009	0.0212 ± 0.0007	0.0205 ± 0.0006
	Time (s)	17.38 ± 0.09	27.81 ± 0.14	45.85 ± 0.36	113.49 ± 0.40	649.99 ± 1.32
	Memory (GB)	0.53 ± 0.00	0.53 ± 0.00	0.54 ± 0.00	1.11 ± 0.01	5.57 ± 0.04
AT-AT	RMSE/atom	0.0110 ± 0.0008	0.0627 ± 0.0529	0.0095 ± 0.0008	0.0096 ± 0.0008	0.0096 ± 0.0007
	Time (s)	5.18 ± 0.02	7.91 ± 0.00	13.97 ± 0.03	33.32 ± 0.24	192.14 ± 0.46
	Memory (GB)	0.18 ± 0.00	0.18 ± 0.00	0.32 ± 0.00	0.90 ± 0.00	5.31 ± 0.05
AT-AT-CG-CG	RMSE/atom	0.0143 ± 0.0008	0.0131 ± 0.0009	0.0129 ± 0.0008	0.0129 ± 0.0008	0.0129 ± 0.0008
	Time (s)	2.94 ± 0.01	4.44 ± 0.03	7.62 ± 0.03	17.67 ± 0.04	98.14 ± 0.31
	Memory (GB)	0.18 ± 0.00	0.19 ± 0.00	0.35 ± 0.00	0.92 ± 0.01	5.19 ± 0.20
Stachyose	RMSE/atom	0.0213 ± 0.0008	0.0193 ± 0.0005	0.0198 ± 0.0008	0.0200 ± 0.0008	0.0199 ± 0.0007
	Time (s)	6.65 ± 0.02	10.31 ± 0.03	18.57 ± 0.05	43.18 ± 0.11	251.72 ± 0.65
	Memory (GB)	0.31 ± 0.00	0.31 ± 0.00	0.43 ± 0.00	1.02 ± 0.01	5.51 ± 0.15
Buckyball-catcher	RMSE/atom	0.0096 ± 0.0057	0.0031 ± 0.0002	0.0029 ± 0.0002	0.0029 ± 0.0002	0.0029 ± 0.0002
	Time (s)	2.21 ± 0.02	3.05 ± 0.00	5.07 ± 0.01	11.58 ± 0.08	60.70 ± 0.23
	Memory (GB)	0.15 ± 0.00	0.17 ± 0.00	0.33 ± 0.00	0.91 ± 0.01	5.80 ± 0.04
DWNT	RMSE/atom	0.0014 ± 0.0001	0.0011 ± 0.0001	0.0011 ± 0.0001	0.0011 ± 0.0001	0.0011 ± 0.0001
	Time (s)	2.04 ± 0.01	2.51 ± 0.01	4.33 ± 0.01	9.36 ± 0.04	51.67 ± 0.13
	Memory (GB)	0.25 ± 0.00	0.31 ± 0.00	0.59 ± 0.00	1.26 ± 0.01	5.88 ± 0.04

memory also reflects the storage of data and other tensors, so its observed growth need not follow the $\mathcal{O}(m^4)$ covariance storage term.

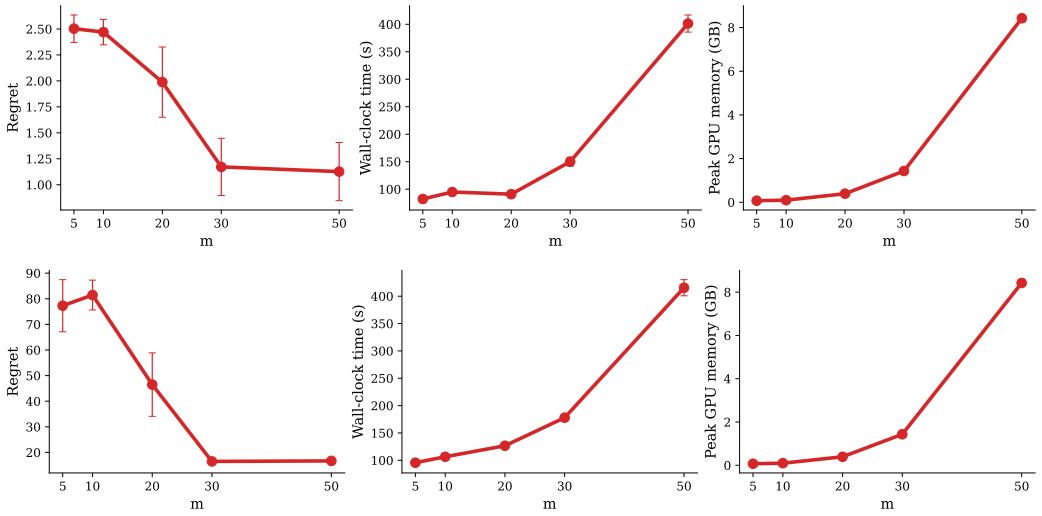

Figure 11: Effect of the Vecchia conditioning set size m on the Ackley-200D (top) and Levy-200D (bottom) benchmarks. The panels show simple regret, end-to-end wall-clock time, and peak GPU memory for TERA with an ARD Matérn-5/2 kernel. Error bars show standard errors across runs.

Figure 11 shows the BO conditioning size sweep on Ackley-200D and Levy-200D. Increasing m from 20 to 30 reduces final mean regret from 1.99 to 1.17 on Ackley-200D and from 46.49 to 16.49 on Levy-200D. Further increasing m to 50 yields little additional improvement on either benchmark. Increasing m gives each local conditional access to more gradient information, but the resulting BO performance also depends on hyperparameter learning, acquisition optimization, and the sequence of selected points. The computational panels show the corresponding tradeoff. Peak GPU memory increases from approximately 0.39 GB at $m = 20$ to 1.43 GB at $m = 30$ and 8.43 GB at $m = 50$, reflecting the growth of the reduced local covariance blocks. Wall-clock time also increases substantially at larger m, reaching approximately 400 seconds at $m = 50$ on both benchmarks. We use $m = 20$ as a common default in the main experiments because it already provides strong performance at low cost, rather than because it is optimal for every task.

E.3 Sensitivity to Observation Noise

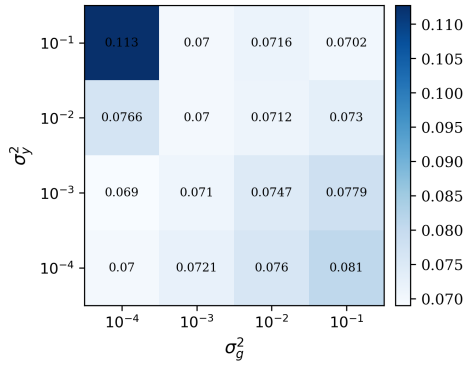

Figure 12: Sensitivity of TERA to the initialization of the observation noise variances for function values, σ_y^2, and gradients, σ_g^2, on the Buckyball-catcher benchmark. Each cell reports raw test RMSE per atom for an isotropic Matérn-5/2 kernel. Lower values are better.

Since the main GP regression experiments initialize the observation-noise parameters to fixed values, we also test the sensitivity of TERA to these initializations. Figure 12 shows how the initial values of the observation noise variances for function values, σ_y^2, and gradients, σ_g^2, affect test accuracy. The heatmap varies σ_y^2 and σ_g^2 over three orders of magnitude and reports the resulting test RMSE per atom. Except for the combination of large function value noise and very small gradient noise, the RMSE values remain in a narrow range across the grid. The main failure mode appears when σ_y^2 is initialized too large while σ_g^2 is initialized too small. This setting overweights gradient information relative to function values at the start of training and gives the largest error in the grid. For moderate or small σ_y^2, the method is substantially less sensitive to σ_g^2.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations in Section 5.4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper provides the assumptions used in the main content and complete proofs in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the experimental details and settings in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets used are all open-access. We provide the code needed to reproduce the experiments at `https://github.com/hseung88/tera`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides the experimental details and settings in Section C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper provides the means and standard errors for experimental results, and figures display error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides the GPU resources in Section 5. All reported wall-clock times and peak GPU memory are measured on a single NVIDIA H100 GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research satisfies the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The datasets and baseline implementations are cited in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code and documentation are publicly available at `https://github.com/hseung88/tera`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.