Accelerated Linearized Laplace Approximation for Bayesian Deep Learning

Zhijie Deng1,2, Feng Zhou2,3, Jun Zhu2,4 ⇤
1 Qing Yuan Research Institute, Shanghai Jiao Tong University
2 Dept. of Comp. Sci. & Tech., BNRist Center, THU-Bosch Joint ML Center, Tsinghua University
3 Center for Applied Statistics, School of Statistics, Renmin University of China
4 Pazhou Laboratory (Huangpu), Guangzhou, China
zhijied@sjtu.edu.cn, feng.zhou@ruc.edu.cn, dcszj@tsinghua.edu.cn

Abstract

Laplace approximation (LA) and its linearized variant (LLA) enable effortless adaptation of pretrained deep neural networks to Bayesian neural networks. The generalized Gauss-Newton (GGN) approximation is typically introduced to improve their tractability. However, LA and LLA are still confronted with non-trivial inefficiency issues and should rely on Kronecker-factored, diagonal, or even last-layer approximate GGN matrices in practical use. These approximations are likely to harm the fidelity of learning outcomes. To tackle this issue, inspired by the connections between LLA and neural tangent kernels (NTKs), we develop a Nyström approximation to NTKs to accelerate LLA. Our method benefits from the capability of popular deep learning libraries for forward mode automatic differentiation, and enjoys reassuring theoretical guarantees. Extensive studies reflect the merits of the proposed method in aspects of both scalability and performance. Our method can even scale up to architectures like vision transformers. We also offer valuable ablation studies to diagnose our method. Code is available at https://github.com/thudzj/ELLA.

1 Introduction

Deep neural networks (DNNs) excel at modeling deterministic relationships and have become de facto solutions for diverse pattern recognition problems [18, 54]. However, DNNs fall short in reasoning about model uncertainty [1] and suffer from poor calibration [17]. These issues are intolerable in risk-sensitive scenarios like self-driving [27], healthcare [34], finance [25], etc.

Bayesian Neural Networks (BNNs) have emerged as effective prescriptions to these pathologies [37, 21, 44, 16]. They usually proceed by estimating the posteriors over high-dimensional NN parameters. Due to some intractable integrals, diverse approximate inference methods have been applied to learning BNNs, spanning variational inference (VI) [1, 36], Markov chain Monte Carlo (MCMC) [55, 4], Laplace approximation (LA) [38, 50, 30], etc.

LA has recently gained unprecedented attention because its post-hoc nature nicely suits with the pretraining-finetuning fashion in deep learning (DL). LA approximates the posterior with a Gaussian around its maximum, whose mean and covariance are the maximum a posteriori (MAP) and the inversion of the Hessian respectively. It is a common practice to approximate the Hessian with the generalized Gauss-Newton (GGN) matrix [40] to make the whole workflow more tractable.

Linearized LA (LLA) [14, 22] applies LA to the first-order approximation of the NN of concern. Immer et al. [22] argue that LLA is more sensible than LA in the presence of GGN approximation;

*Corresponding author. Majority of work completed while Zhijie Deng was at Tsinghua University.

What's more, we theoretically analyze the approximation error between the predictive of ELLA and Analytical estimation is usually intractable due to NNs’ high nonlinearity. Thereby, BNN methods arent all likely interpretations. Typically, BNNs impose a prior \( \pi(\theta) \) on NN parameters and chase the Bayesian posterior \( p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \) where \( p(D|\theta) = \prod_i p(y_i|x_i, \theta) = \prod_i p(y_i|g_\theta(x_i)) \). Analytical estimation is usually intractable due to NNs’ high nonlinearity. Thereby, BNN methods usually find a surrogate of the true posterior \( q(\theta) \approx p(\theta|D) \) via approximate inference methods like variational inference (VI) [1, 20, 36, 62, 29], Laplace approximation (LA) [38, 50], Markov chain Monte Carlo (MCMC) [55, 4, 63], particle-optimization based variational inference (POVI) [35], etc.

BNNs predict for new data \( x_s \) by posterior predictive \( p(y|x_s, D) = E_{q(\theta|D)} p(y|x_s, \theta) \approx E_{q(\theta)} p(y|x_s, \theta) \approx \frac{1}{S} \sum_{s=1}^{S} p(y|g_\theta(x_s)) \) where \( \theta_s \sim q(\theta) \) are i.i.d. Monte Carlo (MC) samples.

LLA can perform on par with or better than popular alternatives on various uncertainty quantification (UQ) tasks [14, 6, 7]. The Laplace library [6] further substantially advances LLA’s applicability, making it a simple and competing baseline for Bayesian DL.

The practical adoption of LA and LLA actually entails further approximations on top of GGN. E.g., when using Laplace to process a pretrained ResNet [18], practitioners are recommended to resort to a Kronecker-factored (KFAC) [40] or diagonal approximation of the full GGN matrix for tractability. An orthogonal tactic is to apply LA/LLA to only NNs’ last layer [30]. Yet, the approximation errors in these cases can hardly be identified, significantly undermining the fidelity of the learning outcomes.

This paper aims at scaling LLA up to make probabilistic predictions in a more assurable way. We first revisit the inherent connections between Neural Tangent Kernels (NTKs) [24] and LLA [28, 22], and find that, if we can approximate the NTKs with the inner product of some low-dimensional vector representations of the data, LLA can be considerably accelerated. Given this finding, we propose to adapt the Nyström method to approximate the NTKs of multi-output NNs, and advocate leveraging forward mode automatic differentiation (fwAD) to efficiently compute the involved Jacobian-vector products (JVPs). The resultant accelerated LLA (ELLA) preserves the principal structures of vanilla LLA yet without explicitly computing/storing the costly GGN/Jacobian matrices for the training data. What’s more, we theoretically analyze the approximation error between the predictive of ELLA and that of vanilla LLA, and find that it deceases rapidly as the Nyström approximation becomes accurate.

We perform extensive studies to show that ELLA can be a low-cost and effective baseline for Bayesian DL. We first describe how to specify the hyperparameters of ELLA, and use an illustrative regression task to demonstrate the effectiveness of ELLA (see Figure 1). We then experiment on standard image classification benchmarks to exhibit the superiority of ELLA over competing baselines in aspects of both performance and scalability. We further show that ELLA can even scale up to modern architectures like vision transformers (ViTs) [11].

2 Background

Consider a learning problem on \( D = (X, Y) = \{(x_i, y_i)\}_{i=1}^N \), where \( x_i \in \mathcal{X} \) and \( y_i \in \mathbb{R}^C \) (e.g., regression) or \( \{0, 1\}^C \) (e.g., classification) refer to observations and targets respectively. The advance in machine learning suggests using an NN \( g_\theta(\cdot) : \mathcal{X} \to \mathbb{R}^C \) with parameters \( \theta \in \mathbb{R}^P \) for data fitting. Despite well-performing, regularly trained NNs only capture the most likely interpretation for the data, thus miss the ability to reason about uncertainty and are prone to overfitting and overconfidence.

BNNs [37, 21, 44] characterize model uncertainty by probabilistic principle and can holistically represent all likely interpretations. Typically, BNNs impose a prior \( \pi(\theta) \) on NN parameters and chase the Bayesian posterior \( p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \) where \( p(D|\theta) = \prod_i p(y_i|x_i, \theta) = \prod_i p(y_i|g_\theta(x_i)) \). Analytical estimation is usually intractable due to NNs’ high nonlinearity. Thereby, BNN methods usually find a surrogate of the true posterior \( q(\theta) \approx p(\theta|D) \) via approximate inference methods like variational inference (VI) [1, 20, 36, 62, 29], Laplace approximation (LA) [38, 50], Markov chain Monte Carlo (MCMC) [55, 4, 63], particle-optimization based variational inference (POVI) [35], etc.

BNNs predict for new data \( x_s \) by posterior predictive \( p(y|x_s, D) = E_{q(\theta|D)} p(y|x_s, \theta) \approx E_{q(\theta)} p(y|x_s, \theta) \approx \frac{1}{S} \sum_{s=1}^{S} p(y|g_\theta(x_s)) \) where \( \theta_s \sim q(\theta) \) are i.i.d. Monte Carlo (MC) samples.
2.1 Laplace Approximation and Its Linearized Variant

Typically, LA builds a Gaussian approximate posterior in the form of \( q(\theta) = \mathcal{N}(\theta; \hat{\theta}, \Sigma) \), where \( \hat{\theta} \) denotes the MAP solution, i.e., \( \hat{\theta} = \arg \max_\theta \log p(D|\theta) + \log p(\theta) \), and \( \Sigma \) is the inversion of the Hessian of the negative log posterior w.r.t. parameters, i.e., \( \Sigma^{-1} = -\nabla^2_{\theta} \log p(D|\theta) + \log p(\theta) |_{\theta=\hat{\theta}} \).

Without loss of generality, we base the following discussion on the isotropic Gaussian prior \( p(\theta) = \mathcal{N}(\theta; 0, \sigma^2_0 \mathbf{I}_p) \), so \( -\nabla^2_{\theta} \log p(\theta) |_{\theta=\hat{\theta}} \) equals to \( \mathbf{I}_p / \sigma_0^2 \).

Due to the intractability of the Hessian of GNNs with massive parameters, it is a common practice to use the symmetric positive and semi-definite (SPSD) GGN matrix as a workaround, i.e.,

\[
\Sigma^{-1} = \sum_i J_\theta(x_i) \Lambda(x_i, y_i) J_\theta(x_i) + \mathbf{I}_p / \sigma_0^2, \tag{1}
\]

where \( J_\theta(x) \triangleq \nabla_\theta g_\theta(x) |_{\theta=\hat{\theta}} \) and \( \Lambda(x, y) \triangleq -\nabla^2_{gg} \log p(y|\theta) |_{g=g_\theta(x)} \).

When concatenating \( \{ J_\theta(x_i) \in \mathbb{R}^{C \times P} \}_{i=1}^N \) as a big matrix \( J_\theta \in \mathbb{R}^{NC \times P} \) and organizing \( \{ \Lambda(x_i, y_i) \in \mathbb{R}^{C \times C} \}_{i=1}^N \) as a block-diagonal matrix \( \Lambda_{X,Y} \in \mathbb{R}^{NC \times NC} \), we have:

\[
\Sigma = \left[ J_\theta \Lambda_{X,Y} J_\theta \mathbf{I}_p / \sigma_0^2 \right]^{-1}. \tag{2}
\]

Yet, LA suffers from underfitting [33]. This is probably because GGN approximation implicitly turns the original model \( g_\theta(x) \) into a linear one \( g_\text{lin}_\theta(x) \) = \( g_\theta(x) + J_\theta(x)(\hat{\theta} - \theta) \) but \( g_\theta(x) \) is still leveraged to make prediction. I.e., there exists a shift between posterior inference and prediction [22]. To mitigate this issue, a proposal is to predict with \( g_\text{lin}_\theta(x) \), giving rise to linearized LA (LLA) [14, 28, 22].

LA and LLA nicely fit the pretraining-finetuning fashion in DL − they can be post-hoc applied to pretrained models where only the GGN matrices require to be estimated and further inverted. By LA and LLA, practitioners can adapt off-the-shelf high performing DNNs to BNNs easily.

LLA has revealed strong results on diverse UQ problems [14, 6, 7]. The Laplace library [6] further advances LLA’s applicability, and evidences LLA is competitive to popular alternatives [63, 32, 39].

Scalability issue The GGN matrix of size \( P \times P \) is still unmanageable in modern DL scenarios, so further approximations sparsifying it are always introduced. The diagonal and KFAC [40] approximations are commonly adopted ones [50, 62], where only a diagonal or block-diagonal structure of the original GNN matrix is preserved. An orthogonal tactic is to concern only a subspace of the high-dimensional parameter space (e.g., the parameter space of the last layer [30]) to reduce the scale of the GGN matrix. However, these strategies sacrifice the fidelity of the learning outcomes as the approximation errors in these cases can hardly be theoretically measured. To this end, we develop accelerated Linearized Laplace Approximation (ELLA) to push the limit of LLA in a more assumable way.

3 Methodology

In this section, we first revisit the relation of LLA to Gaussian processes (GPs) and Neural Tangent Kernels (NTKs) [24]. After that, we reveal how to accelerate LLA by kernel approximation. Based on these findings, we develop an efficient implementation of ELLA using the Nyström method [58].

3.1 The Gaussian Process View of LLA

Integrating \( q(\theta) = \mathcal{N}(\theta; \hat{\theta}, \Sigma) \) with the linear model \( g_\text{lin}_\theta(x) \) actually gives rise to a function-space approximate posterior [14, 28, 22] in the form of \( q(f) = \mathcal{G}_P(f|g_\theta(x), \kappa_{\text{LLA}}(x, x')) \) with

\[
\kappa_{\text{LLA}}(x, x') \triangleq J_\theta(x) \Sigma J_\theta(x')^\top. \tag{3}
\]

By Woodbury matrix identity [60], we have:

\[
\Sigma = \left[ J_\theta \Lambda_{X,Y} J_\theta + \mathbf{I}_P / \sigma_0^2 \right]^{-1} = \sigma_0^2 \left( \mathbf{I}_P + J_\theta \Lambda_{X,Y} J_\theta\Sigma J_\theta \Lambda_{X,Y} J_\theta + \mathbf{I}_P / \sigma_0^2 \right)^{-1} J_\theta \Lambda_{X,Y} J_\theta + \mathbf{I}_P / \sigma_0^2.
\]

It follows that

\[
\kappa_{\text{LLA}}(x, x') = \sigma_0^2 \left( \kappa_{\text{NTK}}(x, x') - \kappa_{\text{NTK}}(x, x') \Lambda_{X,Y} \Lambda_{X,Y} \kappa_{\text{NTK}}(x, x') \right)^{-1} \kappa_{\text{NTK}}(x, x') \tag{5}
\]

\(^1\mathbf{I}_p\) refers to the identity matrix of size \( P \times P \).
where \( \kappa_{\text{NTK}}(x, x') \triangleq J_\theta(x)J_\theta(x')^\top \) denotes the neural tangent kernel (NTK) \([24]\) corresponding to \( g_\theta(x) \). Note that \( \kappa_{\text{NTK}}(x, x') \) is a matrix-valued kernel, with values in the space of \( C \times C \) matrices.

The main challenge then turns into the computation and inversion of the gram matrix \( \kappa_{\text{NTK}}(X, X) \) of size \( NC \times NC \). When either \( N \) or \( C \) is large, the estimation of \( \kappa_{\text{LLA}} \) still suffers from inefficiency issue. To address this, existing work \([22]\) assumes independence among the \( C \) output dimensions to cast \( q(f) \) into \( C \) independent GPs following \([51]\), and randomly subsample \( M \ll N \) data points to form a cheap substitute for the original gram matrix. Despite effective in some cases, these approximations are heuristic, lacking a clear theoretical foundation.

### 3.2 Scale Up LLA by Kernel Approximation

We show that if we can approximate \( \kappa_{\text{NTK}}(x, x') \) with the inner product of some explicit \( C \times K \)-dimensional representations of the data, i.e., \( \kappa_{\text{NTK}}(x, x') \approx \varphi(x)\varphi(x)^\top \) with \( \varphi : \mathcal{X} \rightarrow \mathbb{R}^{C \times K} \), the scalability of the whole workflow can be unleashed.

Concretely, letting \( \varphi_X \in \mathbb{R}^{NC \times K} \) be the concatenation of \( \{\varphi(x_i) \in \mathbb{R}^{C \times K}\}_{i=1}^N \), we have (detailed derivation in Appendix A.2; see also \([9]\))

\[
\kappa_{\text{LLA}}(x, x') \approx \sigma_0^2 \left( \varphi(x)\varphi(x)^\top - \varphi(x)\varphi_X \left( \Lambda_{X, X}^{-1} + \sigma_0^2 + \varphi_X\varphi_X^\top \right)^{-1} \varphi_X \varphi(x)^\top \right)
\]

\[
= \varphi(x) \left[ \sum_i \varphi(x_i)\Lambda(x_i, x_i)\varphi(x_i) + \mathbf{I}_K / \sigma_0^2 \right]^{-1} \varphi(x)^\top \triangleq \kappa_{\text{LLA}}(x, x').
\]

The matrix \( \mathbf{G} \in \mathbb{R}^{K \times K} \) possesses a similar formula to the \( \Sigma \) in the original \( \kappa_{\text{LLA}} \) in Equation (3), yet much smaller. Once having \( \varphi \), it is only required to perform one forward pass of \( g \) and \( \varphi \) for each training data to estimate \( \mathbf{G} \). When \( K \) is reasonably small, e.g., \( < 100 \), it is cheap to invert \( \mathbf{G} \).

### 3.3 Approximate NTKs via the Nyström Method

Typical kernel approximation means include Nyström method \([45, 58]\) and random features-based methods \([48, 49, 61, 43, 15, 9]\). Given that the latter routinely relies on a relatively large number of random features to gain a faithful approximation (which means a large \( K \)), we suggest leveraging the Nyström method, which captures only several principal components of the kernel, to build \( \varphi \).

To comfortably apply the Nyström method, we first rewrite \( \kappa_{\text{NTK}}(x, x') \) as a scalar-valued kernel \( \kappa_{\text{NTK}} ((x, i), (x', i')) = J_\theta(x, i)J_\theta(x', i')^\top \) where \( J_\theta(x, i) \triangleq \nabla_\theta [g_\theta(x)]^{(i)} \mid_{\theta = \theta} : \mathcal{X} \times [C] \rightarrow \mathbb{R}^{1 \times P^2} \) computes the gradient of \( i \)-th output w.r.t. parameters.

By Mercer’s theorem \([42]\),

\[
\kappa_{\text{NTK}} ((x, i), (x', i')) = \sum_{k \geq 1} \mu_k \psi_k(x, i)\psi_k(x', i'),
\]

where \( \psi_k \in L^2(\mathcal{X} \times [C], q)^3 \) refer to the eigenfunctions of the NTK w.r.t. some probability measure \( q \) with \( \mu_k \geq 0 \) as the associated eigenvalues. Based on this, the Nyström method discovers the top-\( K \) eigenvalues as well as the corresponding eigenfunctions for kernel approximation.

By definition, the eigenfunctions can represent the spectral information of the kernel:

\[
\int \kappa_{\text{NTK}} ((x, i), (x', i')) \psi_k(x', i')q(x', i') = \mu_k \psi_k(x, i), \forall k \geq 1,
\]

while being orthonormal under \( q \):

\[
\int \psi_k(x, i)\psi_{k'}(x, i)q(x, i) = 1[k = k'], \forall k, k' \geq 1.
\]

In our case, \( q(x, i) \) can be factorized as the product of the data distribution and a uniform distribution over \( \{1, \ldots, C\} \), which can be trivially sampled from. The Nyström method draws \( M (M \geq K) \) i.i.d. samples \( \tilde{X} = \{(x_1, i_1), \ldots, (x_M, i_M)\} \) from \( q \) to approximate the integration in Equation (8):

\[
\frac{1}{M} \sum_{m=1}^{M} \kappa_{\text{NTK}} ((x, i), (x_m, i_m)) \psi_k(x_m, i_m) = \mu_k \psi_k(x, i), \forall k \in [K].
\]

\(^3\) We use superscript to index a multi-dimensional tensor and \([C]\) represents set of integers from 1 to \( C \).

\(^4\) \( L^2(\cdot, q) \) denotes the space of square-integrable functions w.r.t. \( q \).
Applying this equation to these samples gives rise to:

\[
\frac{1}{M} \sum_{m=1}^{M} \kappa_{\text{NTK}} ((x_{m'}, i_m'), (x_m, i_m)) \psi_k(x_m, i_m) = \mu_k \psi_k(x_{m'}, i_{m'}), \forall k \in [K], m' \in [M],
\]

then we arrive at

\[
\frac{1}{M} K \psi_k \approx \mu_k \psi_k, k \in [K],
\]

where \( K = J_{\theta, X} J_{\theta, X}^T \), with \( J_{\theta, X} \in \mathbb{R}^{M \times P} \) as the concatenation of \( \{ J_{\theta}(x_m, i_m) \in \mathbb{R}^{1 \times P} \}_{m=1}^{M} \), and \( \psi_k = [\psi_k(x_1, i_1), ..., \psi_k(x_M, i_M)]^T \). This implies a scaled eigendecomposition problem of \( K \).

We compute the top-\( K \) eigenvalues \( \lambda_1, ..., \lambda_K \) of matrix \( K \) and record the corresponding orthonormal eigenvectors \( u_1, ..., u_K \). Given the constraint in Equation (9), it is easy to see that:

\[
\mu_k \approx \frac{\lambda_k}{M}, \text{ and } \psi_k(x_m, i_m) \approx \sqrt{M} u_k^{(m)}, m \in [M].
\]

Combining Equation (13) and (10) yields the Nyström approximation of the top-\( K \) eigenfunctions:

\[
\hat{\psi}_k(x, i) = \frac{\sqrt{M}}{\lambda_k} \sum_{m=1}^{M} u_k^{(m)} \kappa_{\text{NTK}} ((x, i), (x_m, i_m)) = \frac{\sqrt{M}}{\lambda_k} J_{\theta}(x, i) J_{\theta, X}^T u_k.
\]

Then, the mapping \( \varphi \) which satisfied \( \kappa_{\text{NTK}}(x, x') \approx \varphi(x) \varphi(x')^T \) can be realized as:

\[
[\varphi(x)]^{(i,k)} = \hat{\psi}_k(x, i) \sqrt{\mu_k} = J_{\theta}(x, i) J_{\theta, X}^T u_k / \sqrt{\lambda_k},
\]

\[
\Rightarrow \varphi(x) = [J_{\theta}(x)v_1, ..., J_{\theta}(x)v_K] \text{ with } v_k = J_{\theta, X}^T u_k / \sqrt{\lambda_k}.
\]

**Connection to sparse approximations of GPs** A recent work [57] has shown that sparse variational Gaussian processes (SVGP) [53, 3] is algebraically equivalent to the Nyström approximation because the Evidence Lower BOnund (ELBO) of the former encompasses the learning objective of the latter. That’s to say, ELLA implicitly deploys a sparse approximation to the original GP predictive of LLA.

**Discussion** In this work, the developed Nyström approximation to NTKs is mainly used for accelerating the computation of the predictive covariance of LLA, while it can also be easily applied to future work on the practical application of NTKs.

### 3.4 Implementation

As shown, the estimation of \( \varphi \) on a data point \( x \) degenerates as \( K \) JVPs, which can be accomplished by invoking \texttt{fwAD} for \( K \) times:

\[
\begin{pmatrix}
(x, \theta, v_k)
\end{pmatrix} \xrightarrow{\text{forward}} \begin{pmatrix}
g_{\theta}(x) \\
J_{\theta}(x)v_k
\end{pmatrix}, \quad k \in [K],
\]

where the model output \( g_{\theta}(x) \) and the JVP \( J_{\theta}(x)v_k \) are simultaneously computed in one single forward pass. Prevalent DL libraries like PyTorch [47] and Jax [2] have already been armed with the capability for \texttt{fwAD}. Algorithm 2 details the procedure of building \( \varphi \) in a PyTorch-like style.

With that, we can trivially compute the posterior \( q(f) \) according to Equation (6), as detailed in Algorithm 1. The procedures for estimating \( G \) and the posterior can both be batched for acceleration.

**Computation overhead** The estimation of \( \varphi \) involves \( M \) forward and backward passes of \( g_{\theta} \) and the eigendecomposition of a matrix of size \( M \times M \). After that, we only need to make \( \{ v_k \}_{k=1}^{K} \) persistent, which amounts to storing \( K \) more NN copies. Estimating \( G^{-1} \) requires scanning the training set once and inverting a matrix of \( K \times K \), which is cheap. The evaluation of \( q(f) \) embodies that of \( \varphi \), i.e., performing \( K \) forward passes under the scope of \texttt{fwAD}. This is similar to other BNNs that perform \( S \) forward passes with different MC parameter samples to estimate the posterior predictive.

### 4 Theoretical Analysis

In this section, we theoretically analyze the approximation error of \( \kappa_{\text{ELLA}}(x, x') \) to \( \kappa_{\text{LLA}}(x, x') \).
We first discuss how to specify the hyperparameters of ELLA, then expose an interesting finding. After that, we compare ELLA to competing baselines to evidence its merits in efficacy and scalability.

Algorithm 1: Build the LLA posterior.

```python
# gθ: NN pre-trained by MAP; (X, Y): training set; C: number of classes
# M, K, σ₀^2: hyper-parameters
def estimate_G(φ, X, Y, K, σ₀^2):
    G = zeros(K, K)
    for (x, y) in (X, Y):
        gₓ, φₓ = φ(x)
        Aₓ,y = hessian(all(gₓ, y), gₓ)
        G += φₓ[Aₓ,y]φₓ
    return G + eye(K)/σ₀^2

def _q_f(φ, G⁻¹, x):
    gₓ, φₓ = φ(x)
    κₓ,x = φₓ(G⁻¹φₓ)
    return κ(x, κₓ,x)

q_f = partial(_q_f, φ, G⁻¹)
```

Algorithm 2: Build φ.

```python
def build_φ(gθ, X, C, M, K):
    def _φ(gθ, C, {vk}k in 1..K, x):
        for k in range(K):
            with full.enable():
                gₓ, jvp = gθ(φₓ, vk, x)
                return gₓ, φ_x

    jθₓ,x = zeros(M, dim(θ))
    for m in range(M):
        x_m = uniform_sample(X)
        i_m = uniform_sample(C)
        jθₓ,x[|m|grad(gθ(x_m)[i_m], θ)]
        return partial(_φ, gθ, C, {vk}k in 1..K)
```

To reduce unnecessary complexity, this section assumes using \( M = K \) i.i.d. MC samples when performing the Nyström method. Then, \( \kappa_{ELLA}(x, x') \) can be reformulated as (details deferred to Appendix A.3)

\[
\kappa_{ELLA}(x, x') = J_θ(x) J_θ(x')^T [J_θ(x)J_θ(x')^T Λ x, Λ y J_θ(x)J_θ(x')^T + J_θ(x)J_θ(x')^T/σ₀^2]^{-1} J_θ(x) J_θ(x')^T.
\]

Thereby, the gap between \( \kappa_{ELLA}(x, x') \) and \( \kappa_{LLA}(x, x') \) can be upper bounded by \( E = \| \Sigma' - \Sigma \| \), where \( \| \cdot \| \) represents the matrix 2-norm (i.e., the spectral norm). In typical cases, \( P \gg M \), thus with high probability \( K = J_θ(x) J_θ(x')^T \approx 0 \). We present an upper bound of \( E \) as follows.

**Theorem 1** (Proof in Appendix A.4). Let \( c_A \) be a finite constant associated with \( Δ \), and \( E' \) the error of Nyström approximation \( \| J_θ(x) J_θ(x')^{-1} - J_θ(x) J_θ(x') \|. \) It holds that

\[
E \leq σ₀^2 c_A E' + σ₀^2.
\]

\( E' \) has been extensively analyzed by pioneering work [12, 5, 26], and we simply adapt the results developed by Drineas and Mahoney [12] to our case. We denote the maximum diagonal entry and the \( M + 1 \) th largest eigenvalue of \( J_θ(x) J_θ(x') \) by \( c_a \) and \( λ_{M+1} \) respectively.

**Theorem 2** (Error bound of Nyström approximation). With probability at least \( 1 - 1 - δ \), it holds that:

\[
E' \leq λ_{M+1} + \frac{NC}{\sqrt{M}} c_A (2 + \log \frac{1}{δ}).
\]

Plugging this back to \( E \), we arrive at the corollary below.

**Corollary 1.** With probability at least \( 1 - δ \), the following bound exists:

\[
E \leq σ₀^2 c_A (λ_{M+1} + \frac{NC}{\sqrt{M}} c_A (2 + \log \frac{1}{δ})) + σ₀^2.
\]

As desired, the upper bound of \( E \) decreases along with the growing of the number of MC samples \( M \).

5 Experiments

We first discuss how to specify the hyperparameters of ELLA, then expose an interesting finding. After that, we compare ELLA to competing baselines to evidence its merits in efficacy and scalability.
We notice that 1) the larger we perform an empirical study to inspect how we simply set works. But also of note that tuning the prior precision cannot fully eliminate overfitting. LLA also shows that tuning the prior precision w.r.t. marginal likelihood can alleviate the overfitting of across settings reflect the error of the Nyström method by (6). Reinspecting Equation (1) and (6), we see, with more training data involved, the covariance in the following experiments unless otherwise stated. The approximation errors of LLA in Figure 2 (c) and Appendix C.2. The architecture contains 2 convolutions and a linear head. Batch normalization [23] and ReLU activation are used. The number of parameters \( P \) is 29,034. Larger architectures or larger \( X \) will render the exact evaluation of \( \kappa_{\text{LLA}} \) unapproachable. We quantify the approximation error of the Nyström method by \( \epsilon_{\text{Nystrom}} \triangleq \| J_{\theta,X} J_{\theta,X}^T (J_{\theta,X} J_{\theta,X}^T)^{-1} - J_{\theta,X} J_{\theta,X}^T \| / \| J_{\theta,X} J_{\theta,X}^T \| \), and that from \( \kappa_{\text{ELLA}} \) to \( \kappa_{\text{LLA}} \) by \( \epsilon_{\text{ELLA}} \triangleq 1 / \| \mathbf{K}_{\text{ELLA}} (x, x) - \mathbf{K}_{\text{LLA}} (x, x) \| / \| \mathbf{K}_{\text{LLA}} (x, x) \| \). We vary \( M \) from 4 to 2000 and \( K \) from 4 to \( M \), and report the approximation errors in Figure 2 (a)-(b). We notice that 1) the larger \( K \) the better; 2) when fixing \( K, \epsilon_{\text{Nystrom}} \) and \( \epsilon_{\text{ELLA}} \) decay as \( M \) grows; 3) \( \epsilon_{\text{Nystrom}} \) decays more rapidly than \( \epsilon_{\text{ELLA}} \), echoing Theorem 1. Given that ELLA needs to store \( K \) vectors of size \( P \), small \( K \) is desired for efficiency. \( K \in [16, 32] \) seems to be a reasonable choice given Figure 2. Besides, Appendix C.1 includes a direct study on how the test NLL of ELLA varies w.r.t. \( M \) and \( K \) on CIFAR-10 [31] benchmark. Given these results, we set \( M = 2000 \) and \( K = 20 \) in the following experiments unless otherwise stated.

ELLA (or LLA) finds a GP posterior, so predicts with a variant of the aforementioned posterior predictive, formulated as \( p(y|x_\ast, D) = \mathbb{E}_{p(f|D)} p(y|f(x_\ast)) \approx \mathbb{E}_{q(f)} p(y|f(x_\ast)) = \mathbb{E}_{f \sim \mathcal{N}(\mu_{ELLA}(x_\ast), \sigma_{ELLA}(x_\ast))} p(y|f) \). In classification tasks, we use 512 MC samples to approximate the last expectation and it is cheap.

### 5.2 The Overfitting Issue of LLA

Reinspecting Equation (1) and (6), we see, with more training data involved, the covariance in LLA, ELLA shrinks and the uncertainty dissipates. However, under ubiquitous model misspecification [41], should the uncertainty vanish that fast? We perform a study with ResNets [18] on CIFAR-10 to seek an answer. Concretely, we randomly subsample \( N \) data from the training set of CIFAR-10, and fit ELLA on them. We depict the test negative log-likelihood (NLL), accuracy, and expected calibration error (ECE) [17] of the deployed ELLA in Figure 2 (c) and Appendix C.2. We also provide the corresponding results of LLA [4] in Appendix C.3. The V-shape NLL curves across settings reflect the overfitting issue of ELLA and LLA∗ (or more generally, LLA). Figure 7 also shows that tuning the prior precision w.r.t. marginal likelihood can alleviate the overfitting of LLA∗ to some extent, which may be the reason why such an issue has not been reported by previous works. But also of note that tuning the prior precision cannot fully eliminate overfitting.

To address the overfitting issue, we advocate performing early stop when fitting ELLA/LLA on big data. Specifically, when iterating over the training data to estimate \( G \) (see Equation (6)), we

\[ \epsilon_{\text{Nystrom}} \triangleq \frac{1}{\| \mathbf{K}_{\text{ELLA}} - \mathbf{K}_{\text{LLA}} \|} \]
Table 1: Comparison on test accuracy (%) ↑, NLL ↓, and ECE ↓ on CIFAR-10. We report the average results over 5 random runs. As the accuracy values of most methods are close, we do not highlight the best.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>NLL</td>
<td>ECE</td>
<td>Acc.</td>
<td>NLL</td>
<td>ECE</td>
<td>Acc.</td>
<td>NLL</td>
<td>ECE</td>
<td>Acc.</td>
<td>NLL</td>
</tr>
<tr>
<td>ELLA</td>
<td>92.5</td>
<td>0.233</td>
<td>0.009</td>
<td>93.5</td>
<td>0.215</td>
<td>0.008</td>
<td>93.9</td>
<td>0.204</td>
<td>0.007</td>
<td>94.4</td>
<td>0.187</td>
</tr>
<tr>
<td>MAP</td>
<td>92.6</td>
<td>0.282</td>
<td>0.036</td>
<td>93.5</td>
<td>0.292</td>
<td>0.041</td>
<td>94.0</td>
<td>0.275</td>
<td>0.039</td>
<td>94.4</td>
<td>0.252</td>
</tr>
<tr>
<td>MFVI-BF</td>
<td>92.7</td>
<td>0.231</td>
<td>0.016</td>
<td>93.5</td>
<td>0.222</td>
<td>0.020</td>
<td>93.9</td>
<td>0.206</td>
<td>0.018</td>
<td>94.4</td>
<td>0.188</td>
</tr>
<tr>
<td>LLA*</td>
<td>92.6</td>
<td>0.269</td>
<td>0.034</td>
<td>93.5</td>
<td>0.259</td>
<td>0.033</td>
<td>94.0</td>
<td>0.237</td>
<td>0.028</td>
<td>94.4</td>
<td>0.213</td>
</tr>
<tr>
<td>LLA*-KFAC</td>
<td>92.6</td>
<td>0.271</td>
<td>0.035</td>
<td>93.5</td>
<td>0.260</td>
<td>0.033</td>
<td>94.0</td>
<td>0.232</td>
<td>0.028</td>
<td>94.4</td>
<td>0.202</td>
</tr>
<tr>
<td>LLA-Diag</td>
<td>92.2</td>
<td>0.728</td>
<td>0.404</td>
<td>92.7</td>
<td>0.755</td>
<td>0.430</td>
<td>92.8</td>
<td>0.778</td>
<td>0.445</td>
<td>92.9</td>
<td>0.843</td>
</tr>
<tr>
<td>LLA-KFAC</td>
<td>92.0</td>
<td>0.852</td>
<td>0.467</td>
<td>91.8</td>
<td>1.027</td>
<td>0.547</td>
<td>91.4</td>
<td>1.091</td>
<td>0.566</td>
<td>89.8</td>
<td>1.174</td>
</tr>
</tbody>
</table>

Figure 3: NLL (Left) and ECE (Right) on CIFAR-10 corruptions for models trained with ResNet-56 architecture. Each box corresponds to a summary of the results across 19 types of skew.

continuously record the NLL of the current ELLA posterior on some validation data, and stop when there is a trend of overfitting. If we cannot access a validation set easily, we can apply strong data augmentation to some training data to form a substitute of the validation set. Compared to tuning the prior, early stopping also helps reduce the time cost of processing big data (e.g., ImageNet [8]).

5.3 Illustrative Regression

We build a regression problem with \( N = 16 \) samples from \( y = \sin 2x + \epsilon, \epsilon \sim \mathcal{N}(0, 0.2) \) as shown in Figure 1. The model is an MLP with 3 hidden layers and tanh activations, and we pretrain it by MAP for 1000 iterations. For ELLA, we set \( M = 16 \) and \( K = 5 \) for efficiency. Unless stated otherwise, we use the interfaces of Laplace [6] to implement LLA, LLA-KFAC, LLA-Diag, and LLA*. The hyperparameters of the competitors are equivalent to those of ELLA except for some dedicated ones like \( M \) and \( K \). It is clear that ELLA delivers a closer approximation to LLA than LLA-Diag and LLA*. We further quantify the quality of the predictive distributions produced by these approximate LLA methods using certain metrics. Considering that in this case, the predictive distribution for one test datum is a Gaussian distribution, we use the KL divergence between the Gaussians yielded by the approximate LLA method and vanilla LLA as a proxy of the approximation error (averaged over a set of test points). The results are reported in Appendix C.4. As shown, LLA-KFAC comes pretty close to LLA. Yet, LLA seems to underestimate in-between uncertainty in this setting, so ELLA seems to be a more reliable (instead of more accurate) approximation than LLA-KFAC. We also highlight the higher scalability of ELLA than LLA-KFAC (see Figure 4(c)), which reflects that ELLA can strike a good trade-off between efficacy and efficiency.

5.4 CIFAR-10 Classification

Then, we evaluate ELLA on CIFAR-10 benchmark using ResNet architectures [18]. We obtain pretrained MAP models from the open source community. Apart from MAP, LLA*, LLA-Diag, LLA-KFAC, we further introduce last-layer LLA with KFAC approximation (LLA*-KFAC) and mean-field VI via Bayesian finetuning [10] (MFVI-BF) as baselines. LLA cannot be directly applied as the GGN matrices are huge. These methods all locate in the family of Gaussian approximate posteriors and are all post-hoc applied to the pretrained models, so the comparisons will be fair. Regarding the setups, we use \( M = 2000 \) and \( K = 20 \) for ELLA; we use 20 MC samples to estimate

\(^{5}\text{Flipout [56] trick is deployed for variance-reduced gradient estimation.}\)

\(^{6}\text{Storing 2000 vectors of size } P \text{ is costly, so we trade time for memory and compute them whenever needed.}\)
Figure 4: (a) Error versus confidence plots for methods trained on CIFAR-10 and tested on CIFAR-10+SVHN. (b) Test NLL of ELLA varies w.r.t. $K$ on CIFAR-10. (c) Comparison on the wall-clock time used for predicting all CIFAR-10 test data (measured on an NVIDIA A40 GPU). The results are obtained with ResNet-20 architecture.

Table 2: Comparison on test accuracy (%) $\uparrow$, NLL $\downarrow$, and ECE $\downarrow$ on ImageNet. We report the average results over 3 random runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>ResNet-18</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>NLL</td>
<td>ECE</td>
<td>Acc.</td>
<td>NLL</td>
<td>ECE</td>
<td>Acc.</td>
<td>NLL</td>
</tr>
<tr>
<td>ELLA</td>
<td>69.8</td>
<td>1.243</td>
<td>0.015</td>
<td>73.3</td>
<td>1.072</td>
<td>0.018</td>
<td>76.2</td>
<td>0.948</td>
</tr>
<tr>
<td>MAP</td>
<td>69.8</td>
<td>1.247</td>
<td>0.026</td>
<td>73.3</td>
<td>1.081</td>
<td>0.035</td>
<td>76.2</td>
<td>0.962</td>
</tr>
<tr>
<td>MFVI-BF</td>
<td>70.3</td>
<td>1.218</td>
<td>0.042</td>
<td>73.7</td>
<td>1.043</td>
<td>0.033</td>
<td>76.1</td>
<td>0.945</td>
</tr>
</tbody>
</table>

We present the comparison on test accuracy, NLL, and ECE in Table 1. As shown, ELLA exhibits superior NLL and ECE across settings. MFVI-BF also gives good NLL. LLA$^*$ and LLA$^*$-KFAC can improve the uncertainty and calibration of MAP, yet underperforming ELLA. LLA-Diag and LLA-KFAC fail for unclear reasons (also reported by [9]), we thus exclude them from the following studies.

We then examine the models on the widely used out-of-distribution (OOD) generalization/robustness benchmark CIFAR-10 corruptions [19] and report the results in Figure 3 and Appendix C.5. It is prominent that ELLA surpasses the baselines in aspects of NLL and ECE at various levels of skew, implying its ability to make conservative predictions for OOD inputs.

We further evaluate the models on the combination of the CIFAR-10 test data and the OOD SVHN test data. The predictions on SVHN are all regarded as wrong due to label shift. We plot the average error rate of the samples with $\leq \tau$ ($\tau \in [0, 1]$) confidence in Figure 4 (a). As shown, ELLA makes less mistakes than the baselines under various confidence thresholds. Figure 4 (b) displays how $K$ impacts the test NLL. We see that $K \in [20, 30]$ can already lead to good performance, reflecting the efficiency of ELLA. Another benefit of ELLA is that with it, we can actively control the performance vs. cost trade-off. Figure 4 (c) shows the comparison on the time used for predicting all CIFAR-10 test data. We note that ELLA is slightly slower than MFVI and substantially faster than non-last-layer LLA methods.

5.5 ImageNet Classification

We apply ELLA to ImageNet classification [8] to demonstrate its scalability. The experiment settings are identical with those for CIFAR-10. We observe that all LLA methods implemented with Laplace would cause out-of-memory (OOM) errors or suffer from very long fitting procedures, thus take only MAP and MFVI-BF as baselines. Table 2 presents the comparison on test accuracy, NLL, and ECE with ResNet architectures. We see that ELLA maintains its superiority in ECE while MFVI-BF can induce higher accuracy and lower NLL. This may be attributed to that the pretrained MAP model lies at a sharp maxima of the true posterior so it is necessary to properly adjust the mean of the Gaussian approximate posterior.
We lastly apply ELLA to ViT-B [11]. We compare ELLA to MAP in Table 3 as all other baselines incur OOM errors or crushing long running time. As shown, ELLA beats MAP in multiple aspects. Figure 5 shows the results of ELLA and MAP on ImageNet corruptions [19]. They are consistent with those for CIFAR-10 corruptions. We reveal by this experiment that ELLA can be a more applicable and scalable method than most of existing BNNs.

### 6 Related Work

LA [38, 50, 14, 30, 22, 6, 7] locally approximates the Bayesian posterior with a Gaussian distribution, analogous to VI with Gaussian variations [16, 1, 36, 52, 46] and SWAG [39]. LA can be applied to pretrained models effortlessly while the acquired posteriors are potentially restrictive (see Section 5.5). VI enjoys higher flexibility yet relies on costly training; BayesAdapter [10] seems to be a remedy to the issue but its accessibility is still lower than LA. SWAG stores a series of SGD iterates to heuristically construct an approximate posterior and is empirically weaker than LA/LLA [6].

Though more expressive approaches like deep ensembles [32] and MCMC [55, 4, 63] can explicitly explore diverse posterior modes, they face limitations in efficiency and scalability. What’s more, it has been shown that LA/LLA can perform on par with or better than deep ensembles and cyclical MCMC on multiple benchmarks [6]. This may be attributed to the un-identified, complicated relationships between the parameters space and the function space of DNNs [59]. Also of note that LA can embrace deep ensembles to capture multiple posterior modes [13].

[9] introduces a general kernel approximation technique using neural networks. By contrast, we focus on leveraging kernel approximation to accelerate Laplace approximation. Thus, the focus of the two works is distinct. Indeed, this work shares a similar idea with the Sec 4.3 in [9] that LLA can be accelerated by kernel approximation. But, except for such an idea, this work differentiates from [9] in aspects like motivations, techniques, implementations, theoretical backgrounds, and applications. Our implementation, theoretical analysis, and some empirical findings are all novel.

### 7 Conclusion

This paper proposes ELLA, a simple, effective, and reliable approach for Bayesian deep learning. ELLA addresses the unreliability issues of existing approximations to LLA and is implemented based on Nyström method. We offer theoretical guarantees for ELLA and perform extensive studies to testify its efficacy and scalability. ELLA currently accounts for only the predictive, and extending it to estimate model evidence for model selection [38] deserves future investigation. Using model evidence to select the number of remaining eigenpairs $K$ for ELLA is also viable.

### Acknowledgments and Disclosure of Funding

This work was supported by the National Key Research and Development Program of China (No. 2017YFA0700904), NSF of China Projects (Nos. 62061136001, 62076145, 62106121, U19B2034, U1811461, U19A2081, 6197222); Beijing NSF Project (No. JQ19016); a grant from Tsinghua Institute for Guo Qiang; and the High Performance Computing Center, Tsinghua University. J.Z was also supported by the XPlorer Prize.
References


**Checklist**

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section Related Work.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section Experiments.
(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
(a) If your work uses existing assets, did you cite the creators? [Yes]
(b) Did you mention the license of the assets? [N/A]
(c) Did you include any new assets either in the supplemental material or as a URL? [No]
(d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]