Scalable Bayesian Low-Rank Adaptation of Large Language Models via Stochastic Variational Subspace Inference

Colin Samplawski[1] **Adam D. Cobb**[1] **Manoj Acharya**[1] **Ramneet Kaur**[1] **Susmit Jha**[1]

[1]Neuro-Symbolic Computing and Intelligence Research Group, Computer Science Laboratory, SRI International

Abstract

Despite their widespread use, large language models (LLMs) are known to hallucinate incorrect information and be poorly calibrated. This makes the uncertainty quantification of these models of critical importance, especially in high-stakes domains, such as autonomy and healthcare. Prior work has made Bayesian deep learning-based approaches to this problem more tractable by performing inference over the low-rank adaptation (LoRA) parameters of a fine-tuned model. While effective, these approaches struggle to scale to larger LLMs due to requiring further additional parameters compared to LoRA. In this work we present **Scala**ble **B**ayesian **L**ow-Rank Adaptation via Stochastic Variational Subspace Inference (ScalaBL). We perform Bayesian inference in an r-dimensional subspace, for LoRA rank r. By repurposing the LoRA parameters as projection matrices, we are able to map samples from this subspace into the full weight space of the LLM. This allows us to learn all the parameters of our approach using stochastic variational inference. Despite the low dimensionality of our subspace, we are able to achieve competitive performance with state-of-the-art approaches while only requiring $\sim$1000 additional parameters. Furthermore, it allows us to scale up to the largest Bayesian LLM to date, with four times as a many base parameters as prior work.

1 INTRODUCTION

The use of large language models (LLMs) have become ubiquitous across many domains ranging from healthcare [Clusmann et al., 2023], scientific discovery Zhang et al. [2024], cyber-physical systems [Cobb et al., 2023], code generation [Jiang et al., 2024], and general everyday use [Anil et al., 2023]. Therefore, ensuring that these models are reliable and trustworthy has never been more vital. However, it is well known that LLMs output incorrect information in the form of "hallucinations" [Huang et al., 2024] and are often poorly calibrated [Zhu et al., 2023, Spiess et al., 2024]. One direction of research aimed at solving these issues considers quantifying the uncertainty of LLM outputs. A variety of post-hoc approaches have been proposed for this task, such as verbalized confidence [Tian et al., 2023, Xiong et al., 2023], quantifying token level uncertainty [Kuhn et al., 2023, Farquhar et al., 2024], or conformal prediction [Kaur et al., 2024].

In contrast, Bayesian deep learning (BDL) provides a principled approach to the uncertainty quantification of deep models. In this family of approaches, uncertainty quantification is performed by directly inferring a distribution over the weights of the model [Gal and Ghahramani, 2016, Blundell et al., 2015, Lakshminarayanan et al., 2017]. Here we estimate a model's predictive uncertainty for a test instance $\mathbf{x}$, denoted $P(\mathbf{y}|\mathbf{x}, \mathcal{D})$, by using Bayes' Rule to marginalize over the parameter posterior distribution, denoted $P(\mathbf{W}|\mathcal{D})$, via the following integral:

$$P(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int P(\mathbf{y}|\mathbf{x}, \mathbf{W})P(\mathbf{W}|\mathcal{D})\mathrm{d}\mathbf{W} \qquad (1)$$

where $\mathcal{D}$ is a training (or fine-tuning) dataset, and $\mathbf{W}$ are the model parameters. However, when scaling such techniques to LLMs, providing a good approximation of this intractable integral becomes increasingly challenging due to the large dimensionality of $\mathbf{W}$. For this reason, recent work has considered performing Bayesian inference over the smaller subset of parameters learned in popular parameter efficient fine-tuning (PEFT) approaches [Fu et al., 2023].

In the widely-used low-rank adaptation (LoRA) technique of Hu et al. [2022], only a small subset of parameters are updated, saving considerable resources compared to updating the entire parameter set, while still enjoying most of the performance of the base model. Conveniently, the low dimensionality of these parameters additionally makes them well

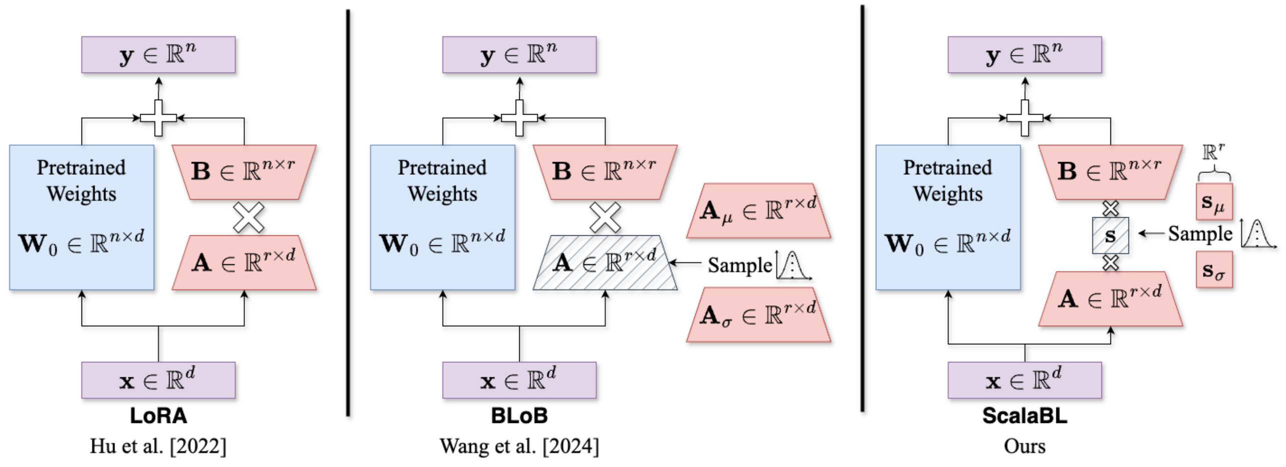
$\mathbf{y} \in \mathbb{R}^n$
Pretrained Weights
$\mathbf{W}_0 \in \mathbb{R}^{n\times d}$
$\mathbf{B} \in \mathbb{R}^{n\times r}$
$\mathbf{A} \in \mathbb{R}^{r\times d}$
$\mathbf{x} \in \mathbb{R}^d$
LoRA
Hu et al. [2022]
$\mathbf{A}_\mu \in \mathbb{R}^{r\times d}$
$\mathbf{A}_\sigma \in \mathbb{R}^{r\times d}$
Sample
BLoB
Wang et al. [2024]
$\mathbb{R}^r$
$\mathbf{s}_\mu$
$\mathbf{s}$
$\mathbf{s}_\sigma$
ScalaBL
Ours

Figure 1: Visual depiction of prior work and our approach for a single layer. Blocks shaded red denote parameters which are trained and blue blocks denote frozen parameters. White blocks with hatching denote parameters which are sampled from learned variational distributions.

suited for BDL techniques. However, Yang et al. [2024a] and Wang et al. [2024] have shown that directly applying BDL techniques such as Deep Ensembles [Lakshminarayanan et al., 2017] or Monte Carlo Dropout [Gal and Ghahramani, 2016] over LoRA only leads to a marginal improvement on uncertainty quantification metrics compared to straightforward fine-tuning approaches such as maximum likelihood estimation (MLE) or Maximum a Posteriori (MAP).

The first success in this space came from Yang et al. [2024a] who perform a Laplace approximation of the parameter posterior after MAP fine-tuning. The state-of-the-art approach of Wang et al. [2024] instead uses stochastic variational inference in a technique they call Bayesian LoRA by Backprop (BLoB). Although this approach performs better than any previous approach, it comes at the cost of needing $\sim 40\%$ more parameters than LoRA. This can be a major memory bottleneck in high-stakes, resource-constrained deployments where computing the Bayesian model average already stresses the available memory budget [Vadera et al., 2022].

In this work, we introduce **Scala**ble **B**ayesian **L**ow Rank Adaptation via Stochastic Variational Subspace Inference (ScalaBL). As shown in Figure 1, we perform Bayesian inference inside a much smaller subspace of the full weight space $\mathbf{W}$ with dimensionality equal to the LoRA rank r. We show how we can repurpose the LoRA parameters $\mathbf{A}$ and $\mathbf{B}$ as projection matrices which map samples from the low dimensional subspace into the full weight space $\mathbf{W}$. We then learn the parameters of our approach using stochastic variational inference.

A major benefit of our approach is that it requires learning only $2r$ additional variational parameters for each LoRA layer, compared to the rd parameters required by BLoB, where d is the embedding dimension of the LLM. For example, when fine-tuning an LLM with 7 billion parameters where $d = 3584$ using a rank of $r = 8$, BLoB requires millions of additional parameters, while ScalaBL requires only ~ 1000. Furthermore, so long as the rank r remains constant, our approach requires the same number of additional parameters per layer regardless of the embedding dimension of the base LLM. As a result, we are able to scale our approach to a 32 billion base parameter model where $d = 5120$, compared to the 7 billion parameter models considered by the prior work of Yang et al. [2024a] and Wang et al. [2024]. Through extensive experimentation, we show that ScalaBL has competitive or superior performance compared to these state-of-the-art baselines on a suite of commonsense reasoning benchmarks in both in- and out-of-distribution settings.

We highlight our main contributions as follows:

- We propose ScalaBL, a Bayesian LoRA approach which performs stochastic variational inference inside a low dimensional subspace.
- ScalaBL enjoys considerable parameter efficiency compared to prior work and requires $\sim 2000\times$ fewer additional parameters.
- ScalaBL achieves competitive or superior performance to state-of-the-art approaches in terms of uncertainty quantification metrics, while requiring fewer parameters.
- Our work is the first to scale a Bayesian LoRA approach to a pre-trained model of 32 billion base parameters, compared to the 7 billion parameter models of prior work.

The structure of the paper is as follows. In Section 2, we discuss relevant prior work that our approach builds on. In Section 3.1, we demonstrate our approach for building a parameter-efficient subspace and in Section 3.2, we discuss how to train a probabilistic model in this subspace using stochastic variational inference. In Section 4, we provide results of our extensive experiments. Finally, in Sections 5 and 6, we discuss limitations and conclude. Additional details and experimental results are included in the Appendix. Our code is available at `github.com/SRI-CSL/BayesAdapt`.

2 PRIOR WORK

In this section we discuss prior work which our approach builds upon.

2.1 LOW-RANK ADAPTATION

The low-rank adaptation (LoRA) approach of Hu et al. [2022] has become a standard technique for fine-tuning LLMs in a tractable way. Consider a linear layer inside a pretrained LLM which has weights $\mathbf{W}_0 \in \mathbb{R}^{n \times d}$, where d is the embedding dimension of the model and n is the output dimension of the layer. A forward pass through the layer for a batch of b input features $\mathbf{x} \in \mathbb{R}^{b \times d}$ is given by $\mathbf{y} = \mathbf{x}\mathbf{W}_0^T$. In LoRA, rather than updating all of the model parameters, $\mathbf{W}_0$, we instead keep these parameters fixed and learn a new pair of low-rank parameters $\mathbf{A} \in \mathbb{R}^{r \times d}$ and $\mathbf{B} \in \mathbb{R}^{n \times r}$, such that:

$$\mathbf{y} = \mathbf{x}\mathbf{W}_0^T + \mathbf{x}(\mathbf{B}\mathbf{A})^T \tag{2}$$

The value $r \ll \min(n, d)$ is commonly known as the LoRA rank. In this way, only $r(n + d)$ parameters need to be learned rather than nd, leading to considerable resource savings with minimal performance penalty.

2.2 LAPLACE LORA

The approach of Yang et al. [2024a] is the first example in the literature of applying uncertainty quantification techniques to LoRA layers by applying a Laplace approximation to the low-rank parameters. They treat a fine-tuned MAP estimate as the mean of a multivariate Gaussian distribution with covariance derived from the inverse Hessian. However, even when restricting the Laplace approximation to the LoRA parameters, evaluation of the Hessian is infeasible. Therefore Yang et al. [2024a] add structure to the Hessian by using a Kronecker factorization [Ritter et al., 2018, Daxberger et al., 2021]. These Kronecker factors are still memory intensive, so Yang et al. [2024a] are forced to perform a further approximation via an iterative truncated singular value decomposition approach. The Laplace approximation is performed post-hoc after fine-tuning the LoRA parameters. An additional limitation is that at test time, they need to backpropagate through the model to build the approximated covariance matrix. This limits the scalability and use of their approach in resource-constrained environments.

2.3 BLOB

The current state-of-the-art approach in this space is Bayesian Low-Rank Adaptation by Backpropagation (BLoB). BLoB moves away from the two stage approach of Laplace LoRA and instead performs stochastic variational inference over the LoRA parameters $\mathbf{A}$. More specifically, they follow the Bayes by Backprop approach introduced by Blundell et al. [2015]. That is, they recast $\mathbf{A}$ as the means of a low rank Gaussian distribution, denoted $\mathbf{A}_\mu$ and learn a set of variance parameters $\mathbf{A}_\sigma$. Using the reparameterization trick [Kingma and Welling, 2013], they project samples from this low rank distribution into the full weight space:

$$\mathbf{W}_t = \mathbf{W}_0 + \mathbf{B}(\mathbf{A}_\mu + \mathbf{A}_\sigma \cdot \boldsymbol{\epsilon}_t) \tag{3}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, 1)$. Wang et al. [2024] show empirically that their approach leads to better performance than Laplace LoRA. However, a notable upside of the Laplace approximation is that it does not require learning any additional parameters. Due to the variance parameters $\mathbf{A}_\sigma$, BLoB requires learning $1.4\times$ more parameters than the base LoRA fine-tuning process. Even for smaller 7 billion parameters models, this can be millions of additional parameters.

2.4 BAYESIAN SUBSPACE INFERENCE

Rather than trying to approximate the parameter posterior directly, Izmailov et al. [2020] purpose to perform Bayesian inference in a much smaller k-dimensional subspace of the full parameter space defined by vectors $\mathbf{z} \in \mathbb{R}^k$. They then learn a model $P(\mathbf{z}|\mathcal{D})$, and a projection matrix $\mathbf{P}$. This allows them to project samples into full weight space:

$$\mathbf{W}_t = \mathbf{W}_0 + \mathbf{P}\mathbf{z}_t \tag{4}$$

$$\mathbf{z}_t \sim P(\mathbf{z}|\mathcal{D}) \tag{5}$$

As highlighted by Izmailov et al. [2020], this model is not a reparameterization of the original parameter posterior as the projection into the full parameter space is not invertible. However, it has the upside that performing Bayesian inference in the subspace enables the use of many common Bayesian inference techniques that would otherwise be intractable. To the best of our knowledge, such subspace inference techniques have never been applied to LLMs.

3 METHODS

In this section we provide the details of our proposed approach. We first show how we construct an r-dimensional subspace of the full weight space $\mathbf{W}$ and then discuss how we train a probabilistic model in this subspace using stochastic variational inference.

Algorithm 1 ScalaBL Training Procedure

```
Require: Pretrained weights $\mathbf{W}_0 \in \mathbb{R}^{n\times d}$, fine-tuning dataset $\mathcal{D}$, prior distribution $P(\mathbf{s})$
Require: Number of training epochs $E$, batch size $B$, learning rate $\eta$
Require: KL divergence weight $\beta$, variance initialization parameter $\rho$
 1: $\mathbf{Z} \sim \mathcal{U}(-\sqrt{\frac{1}{d}}, \sqrt{\frac{1}{d}})$                ▷ Sample random matrix
 2: $\_, \mathbf{s}_\mu, \mathbf{A} \leftarrow \text{SVD}(\mathbf{Z})$                ▷ Initialize using SVD
 3: $\mathbf{B} \leftarrow 0$                ▷ Initialize as in LoRA
 4: $\mathbf{s}_\sigma \sim \mathcal{U}(\frac{\rho}{\sqrt{2}}, \rho)$                ▷ Initialize as in BLoB
 5: for epoch $e \leftarrow 1 \dots E$ do
 6:     for batch $\mathcal{D}_t \sim \mathcal{D}$ do
 7:         $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0,1)$                ▷ Sample noise
 8:         $\mathbf{W}_t \leftarrow \mathbf{W}_0 + \mathbf{B}\,\text{diag}(\mathbf{s}_\mu + \mathbf{s}_\sigma \cdot \boldsymbol{\epsilon}_t)\mathbf{A}$                ▷ Reparameterization trick (Equation 13)
 9:         $\mathcal{L}_t \leftarrow -\frac{1}{B}\log P(\mathcal{D}_t|\mathbf{W}_t) + \beta D_{KL}(\mathcal{N}(\mathbf{s}_\mu, \text{diag}(\mathbf{s}_\sigma))||P(\mathbf{s}))$                ▷ Compute ELBO
10:         $\boldsymbol{\theta} \leftarrow [\mathbf{s}_\mu, \mathbf{s}_{\sigma,}, \mathbf{A}, \mathbf{B}]$                ▷ Collect trainable parameters
11:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{\theta}}$                ▷ Compute gradient and update parameters
12:     end for
13: end for
```

3.1 SUBSPACE CONSTRUCTION

To begin, consider a LoRA layer with rank r, initial weights $\mathbf{W}_0$, and low rank factors $\mathbf{A}$ and $\mathbf{B}$. We would like to generate an r-dimensional subspace defined by vectors $\mathbf{s} \in \mathbb{R}^r$ which can be projected into the full weight space $\mathbf{W}$. We retain $\mathbf{B}$ and use it as a projection matrix as in LoRA, allowing us to focus on building a subspace over $\mathbf{A}$. It is tempting to follow Izmailov et al. [2020] and learn a simple linear projection matrix $\mathbf{P}$, resulting in the following subspace:

$$\mathcal{S}_{lin} = \{\mathbf{W}|\mathbf{W} = \mathbf{W}_0 + \mathbf{BPs}\} \tag{6}$$

However, $\mathbf{A}$ is an $r\times d$ matrix, so the product $\mathbf{Ps}$ would need to be a vector of length rd which then could be reshaped. This means that $\mathbf{P}$ would need to have dimensionality $rd\times r$, and since we would take $\mathbf{P}$ to be a matrix of learnable parameters, this choice of subspace would eliminate any parameter savings of LoRA.

To motivate a more parameter efficient subspace construction, consider the truncated Singular Value Decomposition (SVD) of $\mathbf{A}$:

$$\mathbf{U}\,\text{diag}(\mathbf{s})\mathbf{V} = \text{SVD}(\mathbf{A}) \tag{7}$$

where $\mathbf{s} \in \mathbb{R}^r$ is the vector of singular values, $\mathbf{U} \in \mathbb{R}^{r\times r}, \mathbf{V} \in \mathbb{R}^{r\times d}$ are the left and right singular vectors, and $\text{diag}(\cdot)$ is the diagonal embedding operator. Using $\mathbf{U}$ and $\mathbf{V}$ as projection matrices naturally defines the following subspace:

$$\mathcal{S}_{\text{SVD}} = \{\mathbf{W}|\mathbf{W} = \mathbf{W}_0 + \mathbf{BU}\,\text{diag}(\mathbf{s})\mathbf{V}\} \tag{8}$$

We notice that $\mathbf{BU}$ is a product of linear parameter matrices, which has the same representational power as $\mathbf{B}$ alone. Furthermore since the dimensionality of $\mathbf{V}$ is equal to that of $\mathbf{A}$, we simply rename $\mathbf{V}$ to be $\mathbf{A}$. This results in the following subspace:

$$\begin{aligned}\mathcal{S}_{\text{ScalaBL}} &= \{\mathbf{W}|\mathbf{W} = \mathbf{W}_0 + \mathbf{B}\,\text{diag}(\mathbf{s})\mathbf{A}\} && (9)\\ &= \{\mathbf{W}|\mathbf{W} = \mathbf{f}(\mathbf{s})\} && (10)\end{aligned}$$

where $\mathbf{f}$ is a projection function which is defined for notational convenance. Intuitively, we are repurposeing the LoRA parameters as projection matrices for an r-dimensional subspace that sits "in-between" $\mathbf{A}$ and $\mathbf{B}$.

3.2 VARIATIONAL SUBSPACE INFERENCE

Next we build a probabilistic model in this subspace with data likelihood given by:

$$P(\mathcal{D}|\mathbf{s}) = P(\mathcal{D}|\mathbf{W} = \mathbf{f}(\mathbf{s})) \tag{11}$$

We set our variational approximation over $\mathbf{s}$ as an r-dimensional diagonal Gaussian distribution:

$$q_{\boldsymbol{\theta}}(\mathbf{s}) = \mathcal{N}(\mathbf{s}|\mathbf{s}_\mu, \text{diag}(\mathbf{s}_\sigma)) \tag{12}$$

with mean and variance parameters $\boldsymbol{\theta} = [\mathbf{s}_\mu, \mathbf{s}_\sigma]$. To learn these variational parameters we use stochastic variational inference. At training step t, we use the reparameterization trick [Kingma and Welling, 2013] to generate a sample from $q_{\boldsymbol{\theta}}(\mathbf{s})$ and project into the full weight space:

$$\mathbf{W}_t = \mathbf{W}_0 + \mathbf{B}\,\text{diag}(\mathbf{s}_\mu + \mathbf{s}_\sigma \cdot \boldsymbol{\epsilon}_t)\mathbf{A} \tag{13}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0,1)$. We then maximize the evidence lower bound (ELBO) [Jordan et al., 1999] for each batch $\mathcal{D}_t$:

$$\mathcal{L}_t = \log P(\mathcal{D}_t|\mathbf{W}_t) - \beta D_{KL}(q_\theta(\mathbf{s})||P(\mathbf{s})) \tag{14}$$

Here the first term is the data likelihood under the LLM using the weight sample $\mathbf{W}_t$. The second term regularizes

Dataset	Citation	# Classes	Train Samples	Test Samples
Winogrande-Small (WG-S)	Sakaguchi et al. [2021]	2	0.64K	1.27K
Winogrande-Medium (WG-M)	Sakaguchi et al. [2021]	2	2.56K	1.27K
ARC-Easy (ARC-E)	Clark et al. [2018]	4	2.25K	0.57K
ARC-Challenge (ARC-C)	Clark et al. [2018]	4	1.12K	0.30K
OpenBookQA (OBQA)	Mihaylov et al. [2018]	4	4.96K	0.50K
BoolQ	Clark et al. [2019]	2	2.49K	3.27K
MMLU-Chemistry	Hendrycks et al. [2021]	4	-	0.10K
MMLU-Phyics	Hendrycks et al. [2021]	4	-	0.10K

Table 1: Commonsense Reasoning Datasets used in experiments. We note that the MMLU datasets are used only in the out-of-distribution experiments and therefore have no training samples.

$q_\theta(\mathbf{s})$ against a prior $P(\mathbf{s})$, where β is a scalar hyperparameter which controls the regularization strength [Higgins et al., 2017]. Our full training approach is shown in Algorithm 1.

At test-time, we draw N samples from $q_{\boldsymbol{\theta}}(\mathbf{s})$, project them into the full weight space, and compute a Bayesian model average:

$$\mathbb{E}_{\mathbf{s}_n \sim q_{\boldsymbol{\theta}}(\mathbf{s})}[P(\mathbf{y}|\mathbf{x}, \mathbf{s}_n)] \approx \frac{1}{N}\sum_{n=1}^{N} P(\mathbf{y}|\mathbf{x}, \mathbf{f}(\mathbf{s}_n)) \qquad (15)$$

We maintain $\mathbf{A}$ and $\mathbf{B}$ as learnable parameters as in LoRA. We then additionally need to learn just $2r$ variational parameters $\mathbf{s}_\mu$ and $\mathbf{s}_\sigma$. We note that BLoB can also be cast as a form of subspace inference where the LoRA layer itself defines the subspace of $\mathbf{W}$. This is a much higher dimensional subspace than the one used in ScalaBL, so performing variational inference results in needing to learn rd additional parameters per LoRA layer.

4 EXPERIMENTS

In this section we provide experimental comparisons between ScalaBL, several standard baselines, and current state-of-the-art approaches.

4.1 DATASETS

Following the experimental protocol of Yang et al. [2024a] and Wang et al. [2024] we fine-tune and evaluate our approach using a suite of commonsense reasoning datasets shown in Table 1. These datasets are posed as multiple choice questions. Given an input prompt with a question, we elicit the LLM's softmax distribution over the next token. We then select the logits for each possible answer (e.g. A,B,C,D) and renormalize. In this way, we transform these commonsense reasoning tasks into a classification task. This makes it straightforward to compute standard uncertainty metrics. In particular, we report classification accuracy (ACC), expected calibration error (ECE) [Guo et al., 2017], and the negative log likelihood (NLL) of the correct class. See Appendix Section 7.1 for further details on these metrics.

4.2 BASELINES

We compare against a suite of standard baselines. First we consider the standard LoRA training procedure with and without weight decay regularization (labeled MLE and MAP respectively). Next we compare against the standard BDL baselines of Deep Ensembles [Lakshminarayanan et al., 2017], and Monte Carlo Dropout [Gal and Ghahramani, 2016]. Finally, we present results against the two most recent state-of-the-art approaches: the Laplace approximation approach of Yang et al. [2024a] and BLoB [Wang et al., 2024].

4.3 IMPLEMENTATION DETAILS

We build our approach using the `bayesian-peft` library of Wang et al. [2024]. This provides implementations of the standard baselines as well as BLoB. For the Laplace approximation we use the official code provided by Yang et al. [2024a]. In contrast to Yang et al. [2024a] and Wang et al. [2024], we present results on the newer `Qwen2.5` [Yang et al., 2024b] family of models, rather than the older `Llama-2-7b` model [Touvron et al., 2023] of prior work. For the sake of comparison, results using `Llama-2-7b` are provided in Appendix Section 8.5.

Following Yang et al. [2024a] and Wang et al. [2024], we apply LoRA to the query and value parameters of each self-attention layer as well as the softmax output head of the LLM using rank of $r = 8$. We follow the training procedure and hyperparameters of BLoB. All approaches are trained for 5000 steps using the AdamW optimizer. Training was performed using a batch size of 4 for the 7 billion parameter models and a batch size of 2 for the 32 billion parameter model. In contrast to Wang et al. [2024], we train all approaches using 16-bit precision for the frozen model parameters instead of using 8-bit quantization. The learnable model parameters remain 32-bit. All experiments were

Table 2: In-distribution experiment using `Qwen2.5-7B`. We report the mean and standard deviation of test set performance using 8 training seeds. **Bold** and underlined results denote the best and second best mean performance on each metric/dataset.

Metric	Method	Params (M)	WG-S	ARC-C	ARC-E	WG-M	OBQA	BoolQ
ACC ($\uparrow$)	MLE	3.768	$78.86_{\pm 0.8}$	$\underline{89.53}_{\pm 0.4}$	$95.60_{\pm 0.2}$	$82.30_{\pm 0.7}$	$\underline{92.25}_{\pm 0.9}$	$\underline{89.06}_{\pm 0.2}$
	MAP	3.768	$\underline{78.94}_{\pm 0.8}$	$88.98_{\pm 0.8}$	$95.73_{\pm 0.3}$	$82.09_{\pm 0.7}$	$91.72_{\pm 0.7}$	$89.04_{\pm 0.1}$
	MC-Dropout	3.768	$78.57_{\pm 0.4}$	$89.44_{\pm 0.3}$	$95.77_{\pm 0.4}$	$\underline{82.72}_{\pm 1.0}$	$91.80_{\pm 0.6}$	$88.99_{\pm 0.1}$
	Ensemble	11.305	$\mathbf{79.09}_{\pm 0.4}$	$89.44_{\pm 0.5}$	$95.73_{\pm 0.1}$	$\mathbf{83.23}_{\pm 0.4}$	$\mathbf{92.70}_{\pm 0.6}$	$\mathbf{89.13}_{\pm 0.1}$
	Laplace	3.768	$77.28_{\pm 0.6}$	$85.25_{\pm 1.3}$	$95.34_{\pm 0.5}$	$81.99_{\pm 0.7}$	$91.68_{\pm 0.4}$	$87.77_{\pm 0.1}$
	BLoB	5.403	$78.66_{\pm 0.7}$	$\underline{89.53}_{\pm 0.8}$	$\mathbf{96.54}_{\pm 0.3}$	$82.30_{\pm 0.3}$	$91.72_{\pm 0.7}$	$89.05_{\pm 0.2}$
	ScalaBL (ours)	3.769	$78.64_{\pm 0.4}$	$\mathbf{90.16}_{\pm 0.8}$	$\underline{96.26}_{\pm 0.1}$	$81.42_{\pm 0.3}$	$90.90_{\pm 0.5}$	$88.48_{\pm 0.1}$
ECE ($\downarrow$)	MLE	3.768	$20.14_{\pm 0.9}$	$10.11_{\pm 0.5}$	$4.17_{\pm 0.2}$	$16.10_{\pm 0.6}$	$6.40_{\pm 0.8}$	$3.79_{\pm 0.1}$
	MAP	3.768	$19.99_{\pm 0.9}$	$10.54_{\pm 0.7}$	$4.08_{\pm 0.2}$	$16.42_{\pm 0.8}$	$6.61_{\pm 0.6}$	$3.81_{\pm 0.1}$
	MC-Dropout	3.768	$20.15_{\pm 0.3}$	$10.06_{\pm 0.4}$	$4.01_{\pm 0.3}$	$15.46_{\pm 0.8}$	$6.60_{\pm 0.4}$	$3.88_{\pm 0.1}$
	Ensemble	11.305	$19.06_{\pm 0.4}$	$10.13_{\pm 0.4}$	$3.75_{\pm 0.1}$	$13.65_{\pm 0.8}$	$4.96_{\pm 0.6}$	$2.61_{\pm 0.1}$
	Laplace	3.768	$13.32_{\pm 3.6}$	$37.90_{\pm 2.1}$	$33.80_{\pm 3.8}$	$\underline{4.81}_{\pm 0.7}$	$\mathbf{1.90}_{\pm 0.4}$	$\mathbf{1.18}_{\pm 0.2}$
	BLoB	5.403	$\mathbf{7.88}_{\pm 0.3}$	$\mathbf{4.03}_{\pm 1.0}$	$\mathbf{1.60}_{\pm 0.4}$	$5.08_{\pm 0.4}$	$\underline{2.16}_{\pm 0.5}$	$\underline{1.40}_{\pm 0.3}$
	ScalaBL (ours)	3.769	$\underline{8.88}_{\pm 0.5}$	$\underline{5.03}_{\pm 0.9}$	$\underline{1.78}_{\pm 0.2}$	$\mathbf{3.64}_{\pm 0.2}$	$2.43_{\pm 0.7}$	$1.96_{\pm 0.3}$
NLL ($\downarrow$)	MLE	3.768	$1.94_{\pm 0.3}$	$1.05_{\pm 0.1}$	$0.44_{\pm 0.0}$	$1.20_{\pm 0.1}$	$0.38_{\pm 0.1}$	$0.25_{\pm 0.0}$
	MAP	3.768	$1.88_{\pm 0.2}$	$1.05_{\pm 0.1}$	$0.43_{\pm 0.0}$	$1.27_{\pm 0.2}$	$0.39_{\pm 0.0}$	$0.25_{\pm 0.0}$
	MC-Dropout	3.768	$1.90_{\pm 0.2}$	$1.02_{\pm 0.1}$	$0.43_{\pm 0.0}$	$1.07_{\pm 0.0}$	$0.36_{\pm 0.0}$	$0.25_{\pm 0.0}$
	Ensemble	11.305	$1.33_{\pm 0.1}$	$0.75_{\pm 0.0}$	$0.25_{\pm 0.0}$	$0.74_{\pm 0.0}$	$0.27_{\pm 0.0}$	$\underline{0.24}_{\pm 0.0}$
	Laplace	3.768	$\underline{0.55}_{\pm 0.0}$	$0.80_{\pm 0.1}$	$0.51_{\pm 0.1}$	$0.44_{\pm 0.0}$	$\underline{0.23}_{\pm 0.0}$	$0.29_{\pm 0.0}$
	BLoB	5.403	$\mathbf{0.51}_{\pm 0.0}$	$\mathbf{0.30}_{\pm 0.0}$	$\mathbf{0.10}_{\pm 0.0}$	$\mathbf{0.39}_{\pm 0.0}$	$\mathbf{0.21}_{\pm 0.0}$	$\mathbf{0.23}_{\pm 0.0}$
	ScalaBL (ours)	3.769	$\mathbf{0.51}_{\pm 0.0}$	$\underline{0.31}_{\pm 0.0}$	$\underline{0.11}_{\pm 0.0}$	$\underline{0.40}_{\pm 0.0}$	$\underline{0.23}_{\pm 0.0}$	$\underline{0.24}_{\pm 0.0}$

performed on a single 80GB NVIDIA A100 GPU.

For ScalaBL, we use the same KL weighting schedule as BLoB with an maximum value of $\beta = 0.1$ We do not use the Flipout technique [Wen et al., 2018] that was utilized by BLoB as we found that it did not noticeably effect performance. This simplifies the complexity of the implementation of our approach compared to BLoB. As in BLoB, we use a standard $\mathcal{N}(0, I_r)$ as the prior $P(\mathbf{s})$. We initialize $\mathbf{s}_\mu$ and $\mathbf{A}$ by performing an SVD on a randomly initialized matrix. This is a fast operation due to the low rank nature of the LoRA matrices. Like in BLoB, the variance parameters $\mathbf{s}_\sigma$ were initialized as small uniformly random values. We use a log parametrization for $\mathbf{s}_\sigma$ to ensure the variance remain positive. Following the intuition that $\mathbf{s}_\mu$ is analogous to the singular values of $\mathbf{A}$, we ensure their positivity using a log parametrization as well.

For the variational approaches, BLoB and ScalaBL, we present our main results using $N = 10$ posterior weight samples during evaluation, which Wang et al. [2024] found to give the best performance. The effect of this hyperparameter is explored further in Appendix Section 8.1. Similarly, we perform 10 forward passes for the MC-Dropout baseline. For Deep Ensembles, we use an ensemble size of 3.

4.4 IN-DISTRIBUTION RESULTS

In Table 2 we present test set results for a standard in-distribution setting using the `Qwen2.5-7B` LLM. We first notice that a straightforward MLE fine-tuning approach leads to high accuracy across all datasets, but often overfits as evidenced by the poor ECE results. The MAP result is equivalent to MLE with a weight decay penalty of 10^{-2}, which marginally improves final calibration. We see minor improvements in ECE and NLL when moving to Monte Carlo Dropout and Deep Ensembles, with Deep Ensembles performing the best of the standard baselines, albeit at significantly higher resource cost.

Validating the results of Yang et al. [2024a] and Wang et al. [2024], we see that the more recent state-of-the-art approaches out perform the baselines in terms of ECE and NLL, with minimal reduction in classification accuracy. Furthermore, we see that ScalaBL consistently achieves performance that is competitive with BLoB, and even achieves state-of-the-art performance on Winogrande-Medium dataset in terms of ECE.

Unsurprisingly, BLoB often performs the best out of all methods and regularly outperforms ScalaBL by a small margin. However, BLoB has strictly greater representational power than ScalaBL or Laplace due to its higher parameter count. Compared to MLE, BLoB requires an additional

Table 3: Out-of-distribution experiment using `Qwen2.5-7B`. We report the mean and standard deviation of test set performance using 8 training seeds. **Bold** and $\underline{\text{underlined}}$ results denote the best and second best mean performance on each metric/dataset.

Metric	Method	Params (M)	OBQA	ARC-C	ARC-E	Chemistry	Physics
ACC (↑)	MLE	3.768	$\underline{92.25}_{\pm 0.9}$	$90.88_{\pm 0.7}$	$95.64_{\pm 0.5}$	$53.00_{\pm 1.3}$	$53.00_{\pm 1.5}$
	MAP	3.768	$91.72_{\pm 0.7}$	$90.20_{\pm 0.9}$	$95.53_{\pm 0.6}$	$\underline{53.50}_{\pm 0.9}$	$53.25_{\pm 3.1}$
	MC-Dropout	3.768	$91.80_{\pm 0.6}$	$90.37_{\pm 0.5}$	$95.51_{\pm 0.4}$	$52.75_{\pm 1.3}$	$51.00_{\pm 2.1}$
	Ensemble	11.305	$\mathbf{92.70}_{\pm 0.6}$	$90.84_{\pm 0.6}$	$95.71_{\pm 0.5}$	$53.25_{\pm 1.0}$	$\mathbf{53.88}_{\pm 1.2}$
	Laplace	3.768	$91.68_{\pm 0.4}$	$90.51_{\pm 0.7}$	$95.61_{\pm 0.4}$	$48.75_{\pm 1.8}$	$50.74_{\pm 2.3}$
	BLoB	5.403	$91.72_{\pm 0.7}$	$\mathbf{92.49}_{\pm 0.5}$	$\mathbf{96.07}_{\pm 0.5}$	$\mathbf{54.69}_{\pm 1.4}$	$\underline{53.65}_{\pm 2.8}$
	ScalaBL (ours)	3.769	$90.90_{\pm 0.5}$	$\underline{91.06}_{\pm 1.1}$	$\underline{95.74}_{\pm 0.5}$	$52.60_{\pm 1.8}$	$53.13_{\pm 1.5}$
ECE (↓)	MLE	3.768	$6.40_{\pm 0.8}$	$7.72_{\pm 0.6}$	$3.48_{\pm 0.4}$	$23.29_{\pm 2.2}$	$23.22_{\pm 3.2}$
	MAP	3.768	$6.61_{\pm 0.6}$	$7.89_{\pm 0.9}$	$3.31_{\pm 0.2}$	$22.90_{\pm 1.9}$	$21.52_{\pm 4.4}$
	MC-Dropout	3.768	$6.60_{\pm 0.4}$	$7.63_{\pm 0.8}$	$3.38_{\pm 0.2}$	$23.74_{\pm 1.6}$	$21.61_{\pm 2.1}$
	Ensemble	11.305	$4.96_{\pm 0.6}$	$6.18_{\pm 0.6}$	$2.63_{\pm 0.4}$	$19.49_{\pm 1.4}$	$17.33_{\pm 2.0}$
	Laplace	3.768	$\mathbf{1.90}_{\pm 0.4}$	$4.75_{\pm 0.7}$	$\mathbf{1.99}_{\pm 0.4}$	$\mathbf{14.31}_{\pm 2.1}$	$\mathbf{11.94}_{\pm 4.5}$
	BLoB	5.403	$\underline{2.16}_{\pm 0.5}$	$\underline{4.46}_{\pm 0.5}$	$2.35_{\pm 0.4}$	$\underline{16.21}_{\pm 2.2}$	$16.93_{\pm 2.4}$
	ScalaBL (ours)	3.769	$2.43_{\pm 0.7}$	$\mathbf{4.41}_{\pm 0.7}$	$\underline{1.92}_{\pm 0.4}$	$16.94_{\pm 1.8}$	$\underline{16.29}_{\pm 1.8}$
NLL (↓)	MLE	3.768	$0.38_{\pm 0.1}$	$0.44_{\pm 0.0}$	$0.23_{\pm 0.0}$	$1.53_{\pm 0.1}$	$1.18_{\pm 0.1}$
	MAP	3.768	$0.39_{\pm 0.0}$	$0.46_{\pm 0.0}$	$0.22_{\pm 0.0}$	$1.52_{\pm 0.1}$	$1.19_{\pm 0.1}$
	MC-Dropout	3.768	$0.36_{\pm 0.0}$	$0.43_{\pm 0.0}$	$0.21_{\pm 0.0}$	$1.50_{\pm 0.1}$	$1.19_{\pm 0.0}$
	Ensemble	11.305	$0.27_{\pm 0.0}$	$0.33_{\pm 0.0}$	$0.17_{\pm 0.0}$	$1.29_{\pm 0.0}$	$1.07_{\pm 0.0}$
	Laplace	3.768	$0.23_{\pm 0.0}$	$0.32_{\pm 0.0}$	$\underline{0.15}_{\pm 0.0}$	$\mathbf{1.11}_{\pm 0.0}$	$1.03_{\pm 0.0}$
	BLoB	5.403	$\mathbf{0.21}_{\pm 0.0}$	$\underline{0.28}_{\pm 0.0}$	$\underline{0.15}_{\pm 0.0}$	$1.32_{\pm 0.1}$	$\underline{0.99}_{\pm 0.0}$
	ScalaBL (ours)	3.769	$\underline{0.23}_{\pm 0.0}$	$\mathbf{0.27}_{\pm 0.0}$	$\mathbf{0.14}_{\pm 0.0}$	$\underline{1.26}_{\pm 0.0}$	$\mathbf{0.96}_{\pm 0.0}$

$\sim 1.4\times$ as many parameters, while ScalaBL requires only $\sim 1.0001\times$ as many. For this choice of LLM and rank, this results in BLoB adding ~ 1.6 million parameters on top of MLE, while ScalaBL adds only 912. With that in mind, ScalaBL achieves very competitive performance at much lower cost compared to BLoB. For example, on the ARC-Challenge dataset, BLoB sees $\sim 1.3\times$ better ECE performance than ScalaBL with similar accuracy. However, BLoB requires $1792\times$ more additional parameters than ScalaBL.

4.5 OUT-OF-DISTRIBUTION RESULTS

Next we consider an out-of-distribution experiment where models are trained on the OpenBookQA (OBQA) dataset which consists of grade school level, multiple choice science questions. First we evaluate this tuned model on the ARC datasets, which also consists of grade school level multiple choices, representing a smaller distribution shift. Next we investigate a larger distribution shift by evaluating on the more challenging MMLU-Chemistry and MMLU-Physics datasets which consist of undergraduate level chemistry and physics multiple choice questions, respectively. The results of this experiment for all methods are displayed in Table 3.

We again notice that the recent state-of-the-art approaches outperform the standard baselines in terms of uncertainty quantification with comparable accuracy. We see that all methods experience worse calibration when tested under large distribution shift. We additionally point out the poor accuracy of the Laplace method on the MMLU datasets. We see strong performance of our proposed method, with ScalaBL out competing BLoB and Laplace on several datasets in terms of ECE. Under a both small and large amounts of distribution shift, ScalaBL achieves comparable performance to BLoB across all metrics.

4.6 SCALING TO LARGER MODELS

A limitation of the prior work of Yang et al. [2024a] and Wang et al. [2024] is their use of relatively small LLMs with only 7 billion parameters. This makes it unclear if their experimental conclusions generalize to the much larger model sizes which are currently in use [Anil et al., 2023]. For this reason we consider scaling our approach to the largest Bayesian LLM to date, `Qwen2.5-32B`, with four times as many base parameters as prior work. We conduct the same in-distribution experiments as before and present test set results in Table 4. We note that we do not report results using the Laplace baseline as its post-hoc procedure exceeded the memory availability of our 80GB A100 GPU even when using 8-bit parameters and test time batch size of 1, underscoring the poor scalability of this method.

Table 4: In-distribution experiment using `Qwen2.5-32B`. We report the mean and standard deviation of test set performance using 3 training seeds. **Bold** and underlined results denote the best and second best mean performance on each metric/dataset.

Metric	Method	Params (M)	Datasets					
			WG-S	ARC-C	ARC-E	WG-M	OBQA	BoolQ
ACC (↑)	MLE	9.646	$86.45_{\pm 0.6}$	$93.90_{\pm 0.9}$	$\underline{98.80}_{\pm 0.3}$	$90.90_{\pm 0.3}$	$\mathbf{96.93}_{\pm 0.6}$	$\underline{91.42}_{\pm 0.1}$
	MAP	9.646	$86.73_{\pm 0.8}$	$93.85_{\pm 1.0}$	$\mathbf{98.83}_{\pm 0.4}$	$\underline{91.00}_{\pm 0.1}$	$\mathbf{96.93}_{\pm 0.7}$	$\mathbf{91.47}_{\pm 0.1}$
	MC-Dropout	9.646	$\underline{86.81}_{\pm 0.5}$	$94.18_{\pm 0.4}$	$98.71_{\pm 0.5}$	$90.63_{\pm 0.7}$	$\underline{96.60}_{\pm 0.7}$	$\underline{91.42}_{\pm 0.0}$
	Ensemble	28.938	$\mathbf{86.99}_{\pm 0.4}$	$\mathbf{94.97}_{\pm 0.3}$	$98.65_{\pm 0.3}$	$\mathbf{91.42}_{\pm 0.2}$	$\mathbf{96.93}_{\pm 0.6}$	$91.11_{\pm 0.1}$
	BLoB	14.930	$84.92_{\pm 0.4}$	$\underline{94.07}_{\pm 0.4}$	$98.65_{\pm 0.5}$	$90.71_{\pm 0.3}$	$96.53_{\pm 0.3}$	$90.57_{\pm 0.2}$
	ScalaBL (ours)	9.648	$84.73_{\pm 0.4}$	$93.74_{\pm 0.7}$	$98.65_{\pm 0.2}$	$90.07_{\pm 0.1}$	$96.33_{\pm 0.2}$	$90.99_{\pm 0.1}$
ECE (↓)	MLE	9.646	$12.85_{\pm 0.7}$	$5.88_{\pm 0.8}$	$1.04_{\pm 0.3}$	$7.11_{\pm 0.4}$	$2.18_{\pm 0.4}$	$1.66_{\pm 0.1}$
	MAP	9.646	$12.48_{\pm 0.9}$	$5.97_{\pm 0.7}$	$\underline{0.96}_{\pm 0.4}$	$6.83_{\pm 0.4}$	$\underline{2.03}_{\pm 0.6}$	$1.66_{\pm 0.2}$
	MC-Dropout	9.646	$12.22_{\pm 0.5}$	$5.38_{\pm 0.3}$	$1.22_{\pm 0.2}$	$7.50_{\pm 0.2}$	$2.50_{\pm 0.3}$	$1.50_{\pm 0.1}$
	Ensemble	28.938	$11.20_{\pm 0.6}$	$\mathbf{4.89}_{\pm 0.2}$	$\mathbf{0.98}_{\pm 0.4}$	$\mathbf{5.02}_{\pm 0.1}$	$\mathbf{1.85}_{\pm 0.3}$	$\mathbf{0.74}_{\pm 0.1}$
	BLoB	14.930	$\mathbf{7.49}_{\pm 0.3}$	$5.07_{\pm 0.3}$	$1.11_{\pm 0.3}$	$6.18_{\pm 0.5}$	$2.51_{\pm 0.5}$	$\underline{1.39}_{\pm 0.1}$
	ScalaBL (ours)	9.648	$\underline{10.92}_{\pm 0.3}$	$\underline{5.03}_{\pm 0.6}$	$1.06_{\pm 0.1}$	$\underline{5.91}_{\pm 0.2}$	$2.32_{\pm 0.6}$	$1.40_{\pm 0.2}$
NLL (↓)	MLE	9.646	$1.08_{\pm 0.1}$	$0.49_{\pm 0.1}$	$0.06_{\pm 0.0}$	$0.35_{\pm 0.0}$	$0.13_{\pm 0.0}$	$\underline{0.18}_{\pm 0.0}$
	MAP	9.646	$1.05_{\pm 0.0}$	$0.53_{\pm 0.0}$	$0.06_{\pm 0.0}$	$0.33_{\pm 0.0}$	$0.13_{\pm 0.0}$	$\underline{0.18}_{\pm 0.0}$
	MC-Dropout	9.646	$0.99_{\pm 0.0}$	$0.50_{\pm 0.0}$	$0.07_{\pm 0.0}$	$0.36_{\pm 0.0}$	$0.14_{\pm 0.0}$	$\underline{0.18}_{\pm 0.0}$
	Ensemble	28.938	$0.67_{\pm 0.0}$	$\mathbf{0.30}_{\pm 0.0}$	$\mathbf{0.04}_{\pm 0.0}$	$\mathbf{0.25}_{\pm 0.0}$	$\mathbf{0.11}_{\pm 0.0}$	$\underline{0.18}_{\pm 0.0}$
	BLoB	14.930	$\mathbf{0.44}_{\pm 0.0}$	$0.40_{\pm 0.0}$	$0.06_{\pm 0.0}$	$\underline{0.30}_{\pm 0.0}$	$\underline{0.12}_{\pm 0.0}$	$\mathbf{0.17}_{\pm 0.0}$
	ScalaBL (ours)	9.648	$\underline{0.65}_{\pm 0.0}$	$\underline{0.32}_{\pm 0.0}$	$\underline{0.05}_{\pm 0.0}$	$\underline{0.30}_{\pm 0.0}$	$\underline{0.12}_{\pm 0.0}$	$\underline{0.18}_{\pm 0.0}$

In contrast to earlier results, standard baselines are much more competitive when using a larger base model. We see that even simple techniques, such as MLE or MAP, lead to models which are much better calibrated than their smaller counterparts. This phenomenon has been noticed in prior work [Xiong et al., 2023, Spiess et al., 2024]. Furthermore, we see that Deep Ensembles is often the best performing approach across all three metrics. However, this comes at significantly higher resource usage.

We observe that our proposed approach, ScalaBL, continues to show competitive performance against the baselines, including BLoB. It often performs the second best in terms of ECE and NLL, while experiencing a similar classification performance as BLoB. When using a larger base model, the efficiency and scalability of our method is even more pronounced. Moving from `Qwen2.5-7B` to `Qwen2.5-32B` increases the model's embedding dimension from 3584 to 5120 and adds an additional 12 layers. Since the number of variance parameters in BLoB scales with the embedding dimension, it now requires an additional $\sim$5.2 million parameters compared to MLE. By contrast ScalaBL's additional parameter count scales only with r which was not changed for this larger model. For this reason, ScalaBL only requires adding an additional 2064 parameters. In fact, for this choice of LLM and rank, BLoB requires adding 2560$\times$ more parameters than ScalaBL for similar performance. For this reason we feel that ScalaBL is the only method that is capable of scaling to current frontier models, which are already over a trillion base parameters Anil et al. [2023].

5 LIMITATIONS

Regardless of the parameter efficiency of ScalaBL, computing the Bayesian model average over the projected weight samples has the same runtime cost as BLoB. A limitation that this work shares with Yang et al. [2024a] and Wang et al. [2024] is that we only consider multiple choice classification datasets for evaluation. This underscores the need for uncertainty quantification of open-ended LLM generations in future work.

6 CONCLUSION

In this work we presented **Scala**ble **B**ayesian **L**ow Rank Adaptation via Stochastic Variational Subspace Inference (ScalaBL). We perform Bayesian inference over an r-dimensional subspace and repurpose the $\mathbf{A}$ and $\mathbf{B}$ parameters of a LoRA as projection matrices which map samples from this low dimensional subspace into the full weight space $\mathbf{W}$ of an LLM. We showed how we can learn all the parameters of our approach using stochastic variational inference. Due to the small dimensionality of our subspace, we enjoy considerable parameter efficiency compared to prior work, while still achieving competitive performance with state-of-the-art approaches on a variety of commonsense reasoning benchmarks. For this reason our work is the first to scale a Bayesian LoRA approach to a 32 billion model, while requiring several orders of magnitude fewer additional parameters than prior work.

Acknowledgements

This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-23-C-0519 and HR0011-24-9-0424, and the U.S. Army Research Laboratory under Cooperative Research Agreement W911NF-17-2-0196 and Defense Logistics Agency (DLA) and the Advanced Research Projects Agency for Health (ARPA-H) under Contract Number SP4701-23-C-0073. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force, DARPA, the U.S. Army Research Laboratory, ARPA-H or the United States Government.

References

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 2015.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Jan Clusmann, Fiona R Kolbinger, Hannah Sophie Muti, Zunamys I Carrero, Jan-Niklas Eckardt, Narmin Ghaffari Laleh, Chiara Maria Lavinia Löffler, Sophie-Caroline Schwarzkopf, Michaela Unger, Gregory P Veldhuizen, et al. The future landscape of large language models in medicine. *Communications medicine*, 3(1):141, 2023.

Adam Cobb, Anirban Roy, Daniel Elenius, Frederick Heim, Brian Swenson, Sydney Whittington, James Walker, Theodore Bapty, Joseph Hite, Karthik Ramani, et al. Aircraftverse: a large-scale multimodal dataset of aerial vehicle designs. *Advances in Neural Information Processing Systems*, 36:44524–44543, 2023.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34: 20089–20103, 2021.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 2024.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pages 1169–1179. PMLR, 2020.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37: 183–233, 1999.

Ramneet Kaur, Colin Samplawski, Adam D Cobb, Anirban Roy, Brian Matejek, Manoj Acharya, Daniel Elenius, Alexander Michael Berenbeim, John A. Pavlik, Nathaniel D. Bastian, and Susmit Jha. Addressing uncertainty in llms to enhance reliability in generative ai. In *NeurIPS Safe Generative AI Workshop*, 2024.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Claudio Spiess, David Gros, Kunal Suresh Pai, Michael Pradel, Md Rafiqul Islam Rabin, Amin Alipour, Susmit Jha, Prem Devanbu, and Toufique Ahmed. Calibration and correctness of language models for code. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pages 495–507. IEEE Computer Society, 2024.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Meet Vadera, Jinyang Li, Adam Cobb, Brian Jalaian, Tarek Abdelzaher, and Benjamin Marlin. Ursabench: A system for comprehensive benchmarking of bayesian deep neural network models and inference methods. *Proceedings of Machine Learning and Systems*, 4:217–237, 2022.

Yibin Wang, Haizhou Shi, Ligong Han, Dimitris Metaxas, and Hao Wang. Blob: Bayesian low-rank adaptation by backpropagation for large language models. *Conference on Neural Information Processing Systems*, 2024.

Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *International Conference on Learning Representations*, 2018.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *The Twelfth International Conference on Learning Representations*, 2023.

Adam X Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. In *International Conference on Learning Representations*, 2024a.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024b.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. A comprehensive survey of scientific large language models and their applications in scientific discovery. *arXiv preprint arXiv:2406.10833*, 2024.

Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. On the calibration of large language models and alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9778–9795, 2023.

Appendix

Colin Samplawski[1] **Adam D. Cobb**[1] **Manoj Acharya**[1] **Ramneet Kaur**[1] **Susmit Jha**[1]

[1]Neuro-Symbolic Computing and Intelligence Research Group, Computer Science Laboratory, SRI International

7 ADDITIONAL IMPLEMENTATION DETAILS

7.1 UNCERTAINTY QUANTIFICATION METRICS

In our experiments we report results on two popular metrics for measuring uncertainty quantification: negative log likelihood (NLL) and expected calibration error (ECE). For dataset of N test instances $\mathbf{x}_n$ with correct label y_n and a probabilistic model $P_{\boldsymbol{\theta}}$, NLL is defined as:

$$\text{NLL} = \frac{1}{N}\sum_{n=1}^{N} - \log P_{\boldsymbol{\theta}}(y_n|\mathbf{x}_n) \tag{16}$$

That is, it is the expected negative log probability of the correct class under the model.

ECE measures how a model's confidence aligns with the accuracy of its predictions. It can be computed by binning the predictions by their confidence. We then compute a weighted average of the difference between the accuracy and confidence within each bin:

$$\text{ECE} = \sum_{k=1}^{K} \frac{|B_k|}{N} |\text{acc}(B_k) - \text{conf}(B_k)| \tag{17}$$

where K is the number of bins and B_k is the set of samples in the k-th bin. Following Wang et al. [2024], we use $K = 15$ in all experiments.

7.2 PROMPTS

Following Wang et al. [2024], the prompts used for each dataset are displayed in Table 5.

Table 5: Prompts used for each dataset.

Dataset(s)	Prompt
WG-S, **WG-M**	Select one of the choices that answer the following question: {question} Choices: A. {option1}. B. {option2}. Answer:
ARC-E,**ARC-C** **OBQA**,**MMLU**	Select one of the choices that answer the following question: {question} Choices: A. {option1}. B. {option2}. C. {option3}. D. {option4}. Answer:
BoolQ	Answer the question with only True or False: {question} Context: {passage}.

Method	Model Size	Batch Size	Peak Memory Usage (GB)	Total Training Time (s)
ScalaBL	7B	4	22.79	1050
BLoB	7B	4	23.47	1064
ScalaBL	32B	2	72.36	2560
BLoB	32B	2	74.04	2740

Table 6: Training-time resource usage comparison for Winogrande-Small (WG-S) dataset.

7.3 RUNTIME ANALYSIS

The main efficiency savings gained by ScalaBL over BLoB is the reduction of the number of parameters that need to be learned. This also translates to gains in performance during training. In Table 6 we show the training resource usage for both ScalaBL and BLoB. We see that ScalaBL has lower peak memory usage and trains slightly faster than BLoB.

8 ADDITIONAL EXPERIMENTAL RESULTS

8.1 EFFECT OF NUMBER OF SAMPLES

An important hyperparameter for any variational approach is the number of weight samples N to draw when computing the test time Bayesian model average. Using models fine-tuned on the Winogrande-Small dataset we explore different choices for this hyperparameter for both ScalaBL and BLoB. This is shown for all 3 metrics in Figure 2 (Top). We see that performance across all metrics is saturated around $N = 10$, validating the choice of Wang et al. [2024].

Its important to remember that the number of parameters sampled from the variational distribution is much smaller in ScalaBL as compared to BLoB. In Figure 2 (Bottom) we use a log scale plot to compare how many parameters each method has to draw as we increase the number of samples that are performed.

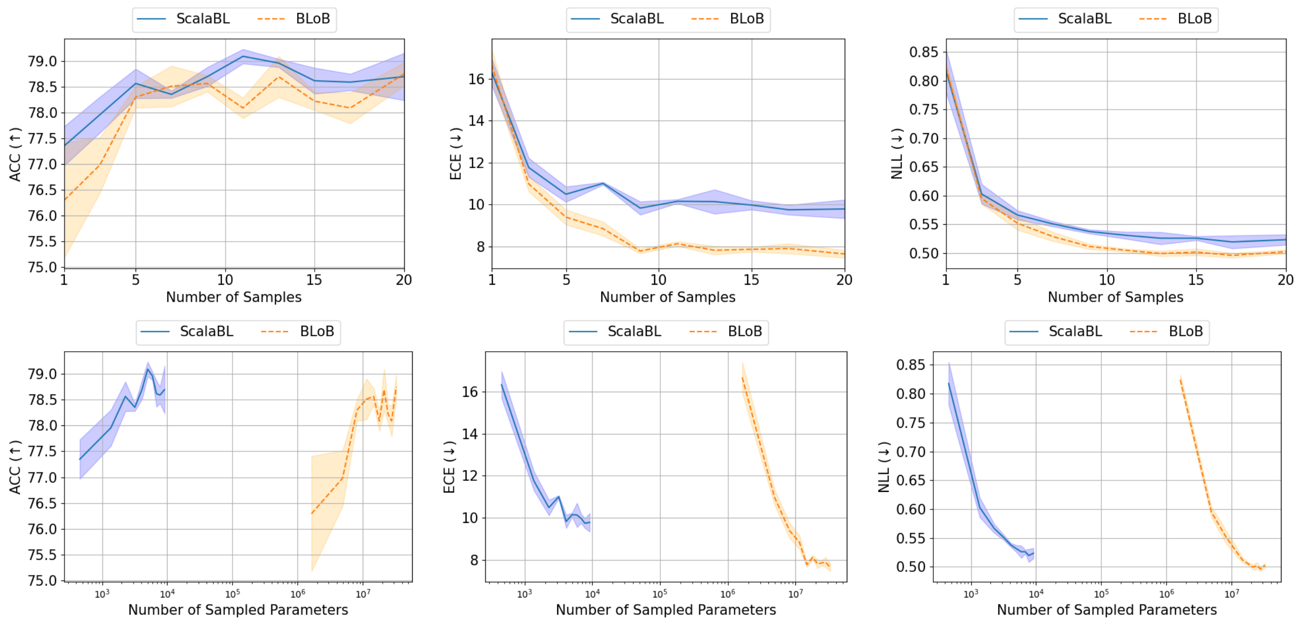

Figure 2: Effect of number of variational samples on Winogrande-Small (WG-S). **Top row:** number of test time samples. **Bottom row:** total number of sampled parameters. Shaded areas show the standard error over three training seeds.

Table 7: In-distribution experiment using `Qwen2.5-7B` using difference choices for the subpace. We report the mean and standard deviation of test set performance using 3 training seeds.

Metric	Subspace	Params (M)	Datasets					
			WG-S	ARC-C	ARC-E	WG-M	OBQA	BoolQ
ACC (↑)	Random	2.135	$73.18_{\pm 0.4}$	$89.30_{\pm 0.2}$	$95.66_{\pm 0.4}$	$74.60_{\pm 0.4}$	$86.80_{\pm 0.6}$	$86.67_{\pm 0.3}$
	SVD	3.773	$78.90_{\pm 0.4}$	$89.30_{\pm 0.5}$	$96.30_{\pm 0.5}$	$81.62_{\pm 0.4}$	$91.07_{\pm 0.5}$	$88.59_{\pm 0.1}$
	ScalaBL	3.769	$78.64_{\pm 0.4}$	$90.16_{\pm 0.8}$	$96.26_{\pm 0.1}$	$81.42_{\pm 0.3}$	$90.90_{\pm 0.5}$	$88.48_{\pm 0.1}$
ECE (↓)	Random	2.135	$12.96_{\pm 0.2}$	$4.75_{\pm 0.7}$	$2.08_{\pm 0.2}$	$6.72_{\pm 0.8}$	$2.89_{\pm 0.4}$	$1.30_{\pm 0.2}$
	SVD	3.773	$9.93_{\pm 0.2}$	$6.01_{\pm 0.5}$	$2.34_{\pm 0.3}$	$5.43_{\pm 0.4}$	$2.57_{\pm 0.6}$	$1.31_{\pm 0.1}$
	ScalaBL	3.769	$8.88_{\pm 0.5}$	$5.03_{\pm 0.9}$	$1.78_{\pm 0.2}$	$3.64_{\pm 0.2}$	$2.43_{\pm 0.7}$	$1.96_{\pm 0.3}$
NLL (↓)	Random	2.135	$0.62_{\pm 0.0}$	$0.31_{\pm 0.0}$	$0.12_{\pm 0.0}$	$0.53_{\pm 0.0}$	$0.37_{\pm 0.0}$	$0.27_{\pm 0.0}$
	SVD	3.773	$0.53_{\pm 0.0}$	$0.35_{\pm 0.0}$	$0.12_{\pm 0.0}$	$0.39_{\pm 0.0}$	$0.22_{\pm 0.0}$	$0.23_{\pm 0.0}$
	ScalaBL	3.769	$0.51_{\pm 0.0}$	$0.31_{\pm 0.0}$	$0.11_{\pm 0.0}$	$0.40_{\pm 0.0}$	$0.23_{\pm 0.0}$	$0.24_{\pm 0.0}$

Table 8: In-distribution experiment using `Qwen2.5-7B` using difference choices for the covariance of $q_{\boldsymbol{\theta}}(\mathbf{s})$. We report the mean and standard deviation of test set performance using 3 training seeds.

Metric	$\boldsymbol{\Sigma}$	Params (M)	Datasets					
			WG-S	ARC-C	ARC-E	WG-M	OBQA	BoolQ
ACC (↑)	Full Rank	3.773	$77.93_{\pm 0.3}$	$89.30_{\pm 0.5}$	$96.48_{\pm 0.2}$	$81.88_{\pm 0.5}$	$91.73_{\pm 0.6}$	$88.74_{\pm 0.1}$
	Diagonal	3.769	$78.64_{\pm 0.4}$	$90.16_{\pm 0.8}$	$96.26_{\pm 0.1}$	$81.42_{\pm 0.3}$	$90.90_{\pm 0.5}$	$88.48_{\pm 0.1}$
ECE (↓)	Full Rank	3.773	$13.25_{\pm 0.6}$	$6.69_{\pm 0.6}$	$2.65_{\pm 0.1}$	$7.11_{\pm 0.2}$	$2.61_{\pm 0.2}$	$1.57_{\pm 0.2}$
	Diagonal	3.769	$8.88_{\pm 0.5}$	$5.03_{\pm 0.9}$	$1.78_{\pm 0.2}$	$3.64_{\pm 0.2}$	$2.43_{\pm 0.7}$	$1.96_{\pm 0.3}$
NLL (↓)	Full Rank	3.773	$0.67_{\pm 0.0}$	$0.41_{\pm 0.0}$	$0.14_{\pm 0.0}$	$0.42_{\pm 0.0}$	$0.23_{\pm 0.0}$	$0.23_{\pm 0.0}$
	Diagonal	3.769	$0.51_{\pm 0.0}$	$0.31_{\pm 0.0}$	$0.11_{\pm 0.0}$	$0.40_{\pm 0.0}$	$0.23_{\pm 0.0}$	$0.24_{\pm 0.0}$

8.2 CHOICE OF SUBSPACE

In this section we consider different choices for the subspace used in method. In Table 7, we present results using the SVD subspace defined in Equation 8. We also include results for an experiment where the $\mathbf{A}$ matrix is frozen during fine-tuning. This is similar to the random subspace approach put forward by Izmailov et al. [2020].

We first notice that difference in performance between the SVD subspace and the subspace used in ScalaBL is negligible. This isn't surprising as adding the extra $\mathbf{U}$ matrix of parameters does not change the expressive power of the model as discussed in the main paper. An interesting upside of using the random subspace is that it further reduces the number of parameters that need to be learned. We see that for some datasets performance is comparable the subspaces with more parameters. However, on some datasets (such as Winogrande-Medium) there is a considerable reduction in classification accuracy when using a random subspace.

8.3 USING A FULL RANK COVARIANCE

Following the prior work, we only considered using a diagonal covariance for $q_{\boldsymbol{\theta}}(\mathbf{s})$ in our experiments in the main paper. For the approaches of Yang et al. [2024a] and Wang et al. [2024] this is a necessary limitation as instantiating a full rank covariance with millions of dimensions would be intractable. However, the Gaussian distribution used in ScalaBL is only r-dimensional. This makes it straightforward to consider using a full rank Gaussian by adding a few more parameters.

We parameterize a full rank covariance matrix $\boldsymbol{\Sigma}$ as an eigen decomposition. We treat the eigenvalues $\mathbf{e} \in \mathbb{R}^r$ and matrix of eigenvectors $\mathbf{E} \in \mathbb{R}^{r \times r}$ as learnable parameters. This adds $r + r^2$ additional parameters to the approach. We use the QR factorization to ensure that eigenvalues are orthogonal.

$$\mathbf{E}, \mathbf{R} = \text{QR}(\hat{\mathbf{E}}) \tag{18}$$

$$\mathbf{\Sigma} = \mathbf{E}\,\text{diag}(\mathbf{e})\mathbf{E}^T \tag{19}$$

where $\hat{\mathbf{E}}$ are free parameters.

We then update the reparameterization trick to use the Cholesky factor of the covariance matrix. We apply the Cholesky factorization on-the-fly during learning.

$$\mathbf{L} = \text{Cholesky}(\mathbf{\Sigma}) \tag{20}$$

$$\mathbf{W}_t = \mathbf{W}_0 + \mathbf{B}\,\text{diag}(\mathbf{s}_\mu + \mathbf{L}\boldsymbol{\epsilon}_t)\mathbf{A} \tag{21}$$

We compare using a diagonal and a full rank covariance matrix in Table 8. We see that using a full rank covariance leads to very similar performance to using a diagonal one, with some datasets even exhibiting worse calibration.

8.4 EFFECT OF LORA RANK

In the prior work of Yang et al. [2024a] and Wang et al. [2024] a LoRA rank of $r = 8$ was used in all experiments. In the main paper, we use this value for the rank as well. In this section we explore the effect of the LoRA on performance for both ScalaBL and BLoB. We ran additional in-distribution experiments using Qwen2.5-7B with $r = [4, 16, 32]$ to compare against the results for $r = 8$ which are already shown in Table 2. This results are shown in Figures 3 and 4.

We first note that the x-axes of these plots show the number of total model parameters, rather than the rank. This captures the fact that BLoB's additional parameters grow more quickly as r increases ($O(rd)$) compared to ScalaBL ($O(r)$). We see that BLoB often results in noticeable drops in accuracy as r increases across multiple datasets (WG-S, ARC-E, ARC-C, WG-M, OBQA). This is then accompanied by increases in NLL. By comparison ScalaBL sees small increases in accuracy across most datasets as r increases, albeit accompanied with small increases in ECE. Furthermore, BLoB sees larger increases in ECE compared to ScalaBL across multiple datasets (ARC-E, ARC-C, OBQA). The only time that BLoB sees a reduction in ECE is when the accuracy also decreases significantly (WG-S, WG-M). ScalaBL is robust to changes in r across all 3 metrics.

8.5 LLAMA2 RESULTS

For the sake of comparison with Yang et al. [2024a] and Wang et al. [2024], we present experimental results using the older `Llama-2-7b` LLM in Tables 9 and 10. We note that we reran BLoB and the standard baselines using 16-bit frozen parameters, instead of 8-bit quantized weights. The reported results for Laplace and Bayes By Backprop (BBB) [Blundell et al., 2015] are repeated from the tables of Wang et al. [2024]. We obverse the same general trends as seen with `Qwen2.5-7B`. Our proposed approach achieves competitive performance with BLoB while requiring significantly fewer parameters.

Table 9: In-distribution experiment using `Llama-2-7b`. We report the mean and standard deviation of test set performance using 3 training seeds. **Bold** and underlined results denote the best and second best mean performance on each metric/dataset.

Metric	**Method**	**Params (M)**	**Datasets**					
			WG-S	**ARC-C**	**ARC-E**	**WG-M**	**OBQA**	**BoolQ**
ACC (↑)	MLE	4.483	70.78$_{\pm 1.2}$	70.16$_{\pm 2.8}$	86.97$_{\pm 0.6}$	75.26$_{\pm 0.6}$	82.53$_{\pm 0.4}$	88.02$_{\pm 0.2}$
	MAP	4.483	71.10$_{\pm 1.1}$	68.81$_{\pm 0.2}$	86.62$_{\pm 0.5}$	**76.45**$_{\pm 0.7}$	82.80$_{\pm 0.2}$	88.05$_{\pm 0.2}$
	MC-Dropout	4.483	69.99$_{\pm 2.7}$	68.13$_{\pm 1.0}$	87.68$_{\pm 0.4}$	76.05$_{\pm 0.7}$	83.07$_{\pm 1.2}$	**88.43**$_{\pm 0.3}$
	Ensemble	13.449	**71.31**$_{\pm 0.3}$	**71.17**$_{\pm 1.4}$	**88.32**$_{\pm 0.3}$	76.37$_{\pm 0.8}$	**83.53**$_{\pm 0.2}$	87.87$_{\pm 0.2}$
	BBB	6.613	56.54$_{\pm 0.7}$	68.13$_{\pm 1.3}$	85.86$_{\pm 0.7}$	73.63$_{\pm 2.4}$	82.06$_{\pm 0.6}$	87.21$_{\pm 0.2}$
	Laplace	4.483	69.20$_{\pm 1.5}$	66.78$_{\pm 0.7}$	80.05$_{\pm 0.2}$	75.55$_{\pm 0.4}$	82.12$_{\pm 0.7}$	86.95$_{\pm 0.1}$
	BLoB	6.613	64.93$_{\pm 5.1}$	70.02$_{\pm 0.9}$	85.80$_{\pm 0.6}$	73.71$_{\pm 1.4}$	82.47$_{\pm 0.4}$	87.62$_{\pm 0.2}$
	ScalaBL (ours)	4.488	70.23$_{\pm 0.9}$	68.58$_{\pm 1.8}$	86.80$_{\pm 0.5}$	74.45$_{\pm 0.9}$	82.13$_{\pm 0.2}$	86.50$_{\pm 0.2}$
ECE (↓)	MLE	4.483	28.17$_{\pm 1.5}$	28.26$_{\pm 1.9}$	12.03$_{\pm 0.9}$	21.97$_{\pm 0.6}$	13.86$_{\pm 0.5}$	4.57$_{\pm 0.3}$
	MAP	4.483	27.77$_{\pm 1.4}$	29.80$_{\pm 0.3}$	11.82$_{\pm 0.1}$	21.08$_{\pm 0.3}$	13.91$_{\pm 0.3}$	4.34$_{\pm 0.2}$
	MC-Dropout	4.483	28.55$_{\pm 2.5}$	29.43$_{\pm 1.1}$	11.25$_{\pm 0.3}$	20.69$_{\pm 0.6}$	12.94$_{\pm 1.2}$	4.17$_{\pm 0.1}$
	Ensemble	13.449	24.61$_{\pm 0.5}$	25.10$_{\pm 1.3}$	10.02$_{\pm 0.1}$	16.96$_{\pm 0.6}$	10.81$_{\pm 0.2}$	3.05$_{\pm 0.2}$
	BBB	6.613	21.81$_{\pm 13.0}$	26.23$_{\pm 1.5}$	12.28$_{\pm 0.6}$	15.76$_{\pm 4.7}$	11.38$_{\pm 1.1}$	3.74$_{\pm 0.1}$
	Laplace	4.483	**4.15**$_{\pm 1.12}$	16.25$_{\pm 2.6}$	33.29$_{\pm 0.6}$	7.40$_{\pm 0.3}$	8.70$_{\pm 1.8}$	1.30$_{\pm 0.3}$
	BLoB	6.613	5.55$_{\pm 3.3}$	14.05$_{\pm 0.7}$	3.39$_{\pm 1.0}$	**3.36**$_{\pm 0.5}$	**2.80**$_{\pm 0.5}$	**1.08**$_{\pm 0.2}$
	ScalaBL (ours)	4.488	9.49$_{\pm 1.2}$	**9.79**$_{\pm 1.9}$	**3.54**$_{\pm 0.2}$	4.31$_{\pm 0.4}$	3.62$_{\pm 0.9}$	1.83$_{\pm 0.3}$
NLL (↓)	MLE	4.483	2.47$_{\pm 0.2}$	2.71$_{\pm 0.4}$	0.98$_{\pm 0.1}$	1.14$_{\pm 0.1}$	0.91$_{\pm 0.1}$	0.27$_{\pm 0.0}$
	MAP	4.483	2.89$_{\pm 0.5}$	3.02$_{\pm 0.2}$	1.05$_{\pm 0.0}$	1.14$_{\pm 0.0}$	0.89$_{\pm 0.0}$	0.27$_{\pm 0.0}$
	MC-Dropout	4.483	3.01$_{\pm 0.4}$	3.08$_{\pm 0.1}$	1.00$_{\pm 0.1}$	1.03$_{\pm 0.0}$	0.86$_{\pm 0.1}$	0.27$_{\pm 0.0}$
	Ensemble	13.449	1.47$_{\pm 0.0}$	1.93$_{\pm 0.1}$	0.74$_{\pm 0.0}$	0.73$_{\pm 0.0}$	0.64$_{\pm 0.0}$	0.25$_{\pm 0.0}$
	BBB	6.613	1.40$_{\pm 0.6}$	2.23$_{\pm 0.0}$	0.91$_{\pm 0.0}$	0.84$_{\pm 0.2}$	0.66$_{\pm 0.1}$	0.31$_{\pm 0.0}$
	Laplace	4.483	**0.60**$_{\pm 0.0}$	1.03$_{\pm 0.0}$	0.88$_{\pm 0.0}$	0.57$_{\pm 0.0}$	0.52$_{\pm 0.0}$	0.31$_{\pm 0.0}$
	BLoB	6.613	0.66$_{\pm 0.1}$	0.87$_{\pm 0.0}$	**0.38**$_{\pm 0.0}$	**0.51**$_{\pm 0.0}$	**0.47**$_{\pm 0.0}$	**0.23**$_{\pm 0.0}$
	ScalaBL (ours)	4.488	0.59$_{\pm 0.0}$	0.79$_{\pm 0.0}$	0.39$_{\pm 0.0}$	**0.51**$_{\pm 0.0}$	0.51$_{\pm 0.0}$	0.25$_{\pm 0.0}$

Table 10: Out-of-distribution experiment using `Llama-2-7b`. We report the mean and standard deviation of test set performance using 3 training seeds. **Bold** and underlined results denote the best and second best mean performance on each metric/dataset.

Metric	**Method**	**Params (M)**	**Datasets**				
			In Dist.	**Smaller Dist. Shift**		**Larger Dist. Shift**	
			OBQA	**ARC-C**	**ARC-E**	**Chemistry**	**Physics**
ACC (↑)	MLE	4.483	$82.53_{\pm 0.4}$	$\underline{69.48}_{\pm 0.5}$	$75.59_{\pm 1.2}$	$39.33_{\pm 1.5}$	$29.00_{\pm 2.6}$
	MAP	4.483	$\underline{82.80}_{\pm 0.2}$	$68.92_{\pm 1.2}$	$76.29_{\pm 0.7}$	$36.00_{\pm 1.0}$	$\underline{31.00}_{\pm 1.0}$
	MC-Dropout	4.483	$83.07_{\pm 1.2}$	$69.14_{\pm 0.5}$	$76.17_{\pm 0.9}$	$37.67_{\pm 2.1}$	$28.00_{\pm 4.4}$
	Ensemble	13.449	$\mathbf{83.53}_{\pm 0.2}$	$69.37_{\pm 0.5}$	$76.12_{\pm 1.0}$	$38.33_{\pm 1.5}$	$29.00_{\pm 2.6}$
	BBB	6.613	$82.06_{\pm 0.6}$	$67.25_{\pm 1.2}$	$75.83_{\pm 0.8}$	$42.36_{\pm 0.5}$	$30.21_{\pm 2.3}$
	Laplace	4.483	$82.12_{\pm 0.7}$	$69.14_{\pm 1.2}$	$74.94_{\pm 1.0}$	$\mathbf{44.10}_{\pm 1.3}$	$\mathbf{31.60}_{\pm 0.5}$
	BLoB	6.613	$82.47_{\pm 0.4}$	$\mathbf{69.56}_{\pm 1.1}$	$\underline{76.55}_{\pm 0.3}$	$\underline{43.40}_{\pm 0.6}$	$30.56_{\pm 1.2}$
	ScalaBL (ours)	4.484	$82.13_{\pm 0.2}$	$\underline{69.48}_{\pm 0.5}$	$\mathbf{77.46}_{\pm 0.3}$	$42.00_{\pm 2.6}$	$30.33_{\pm 0.6}$
ECE (↓)	MLE	4.483	$13.86_{\pm 0.5}$	$23.07_{\pm 0.9}$	$17.41_{\pm 0.9}$	$22.56_{\pm 2.5}$	$29.36_{\pm 2.3}$
	MAP	4.483	$13.91_{\pm 0.3}$	$24.10_{\pm 0.9}$	$16.93_{\pm 1.0}$	$25.96_{\pm 2.0}$	$28.30_{\pm 2.5}$
	MC-Dropout	4.483	$12.94_{\pm 1.2}$	$23.44_{\pm 0.7}$	$16.84_{\pm 0.7}$	$23.78_{\pm 3.0}$	$32.71_{\pm 4.0}$
	Ensemble	13.449	$10.81_{\pm 0.2}$	$19.12_{\pm 1.1}$	$13.66_{\pm 0.9}$	$15.94_{\pm 1.5}$	$\underline{20.86}_{\pm 2.5}$
	BBB	6.613	$11.38_{\pm 1.1}$	$19.90_{\pm 0.7}$	$13.41_{\pm 0.9}$	$15.67_{\pm 1.2}$	$26.10_{\pm 4.8}$
	Laplace	4.483	$8.70_{\pm 1.8}$	$\mathbf{5.84}_{\pm 0.6}$	$\underline{8.51}_{\pm 1.1}$	$\mathbf{10.76}_{\pm 3.4}$	$\mathbf{13.91}_{\pm 0.9}$
	BLoB	6.613	$\mathbf{2.80}_{\pm 0.5}$	$13.82_{\pm 0.5}$	$9.65_{\pm 0.7}$	$\underline{15.39}_{\pm 3.4}$	$22.66_{\pm 0.7}$
	ScalaBL (ours)	4.484	$\underline{3.62}_{\pm 0.9}$	$\underline{11.85}_{\pm 0.6}$	$\mathbf{7.89}_{\pm 0.8}$	$15.99_{\pm 3.3}$	$21.98_{\pm 1.1}$
NLL (↓)	MLE	4.483	$0.91_{\pm 0.1}$	$1.42_{\pm 0.1}$	$1.11_{\pm 0.1}$	$1.62_{\pm 0.0}$	$1.69_{\pm 0.1}$
	MAP	4.483	$0.89_{\pm 0.0}$	$1.46_{\pm 0.1}$	$1.12_{\pm 0.0}$	$1.67_{\pm 0.1}$	$1.70_{\pm 0.1}$
	MC-Dropout	4.483	$0.86_{\pm 0.1}$	$1.39_{\pm 0.1}$	$1.12_{\pm 0.1}$	$1.64_{\pm 0.1}$	$1.76_{\pm 0.0}$
	Ensemble	13.449	$0.64_{\pm 0.0}$	$1.03_{\pm 0.0}$	$0.82_{\pm 0.0}$	$1.42_{\pm 0.0}$	$1.49_{\pm 0.0}$
	BBB	6.613	$0.66_{\pm 0.1}$	$1.06_{\pm 0.0}$	$0.79_{\pm 0.0}$	$1.49_{\pm 0.0}$	$1.62_{\pm 0.1}$
	Laplace	4.483	$0.52_{\pm 0.0}$	$\mathbf{0.81}_{\pm 0.0}$	$\underline{0.70}_{\pm 0.0}$	$\mathbf{1.35}_{\pm 0.0}$	$\mathbf{1.36}_{\pm 0.0}$
	BLoB	6.613	$\mathbf{0.47}_{\pm 0.0}$	$0.88_{\pm 0.0}$	$\underline{0.70}_{\pm 0.0}$	$\underline{1.38}_{\pm 0.0}$	$\underline{1.43}_{\pm 0.0}$
	ScalaBL (ours)	4.484	$\underline{0.51}_{\pm 0.0}$	$\underline{0.85}_{\pm 0.0}$	$\mathbf{0.63}_{\pm 0.0}$	$1.40_{\pm 0.0}$	$1.48_{\pm 0.0}$

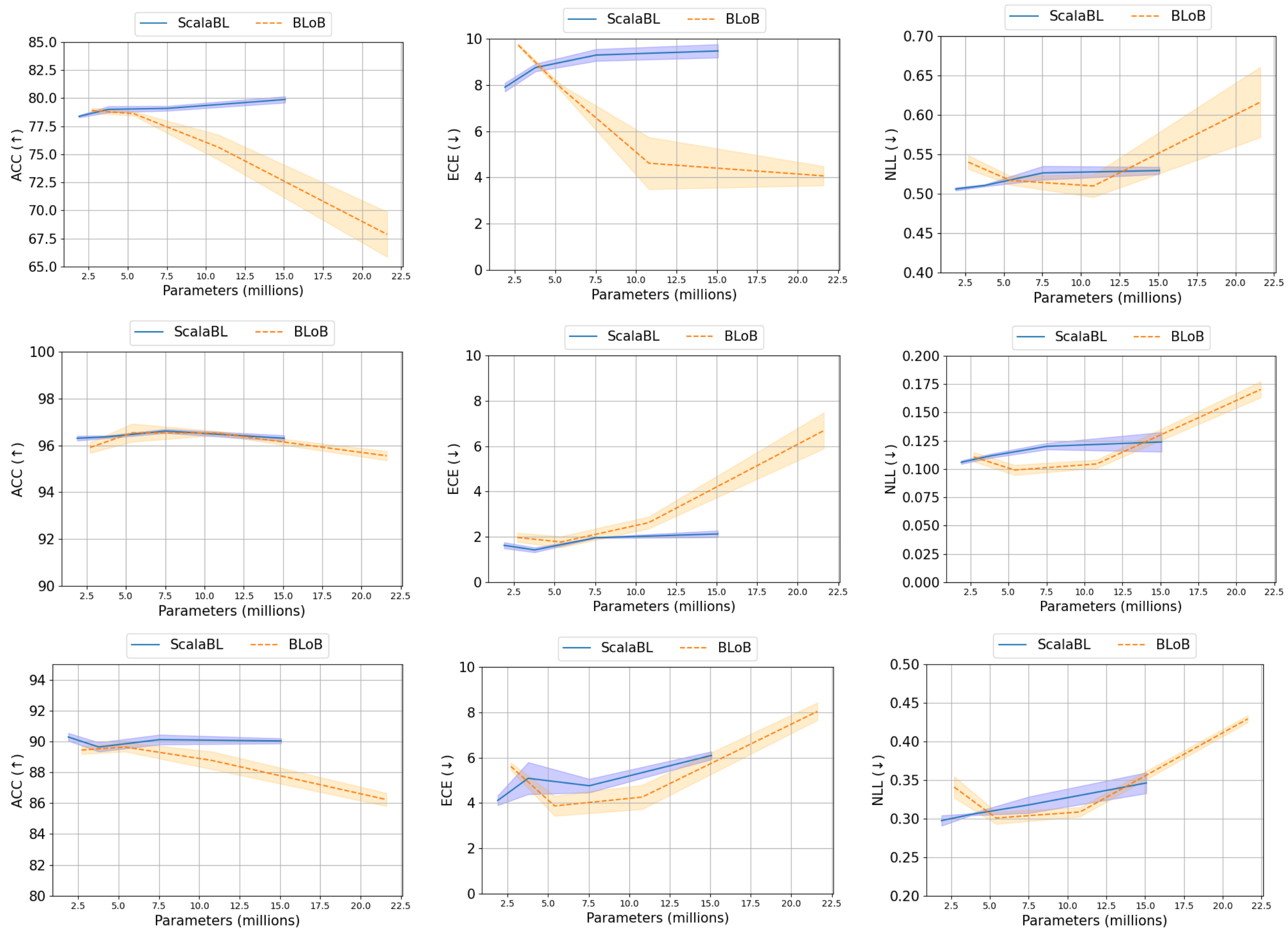

Figure 3: Effect of LoRA rank r. **Top:** Winogrande-Small (WG-S). **Middle:** ARC-Easy (ARC-E). **Bottom:** ARC-Challenge (ARC-C).

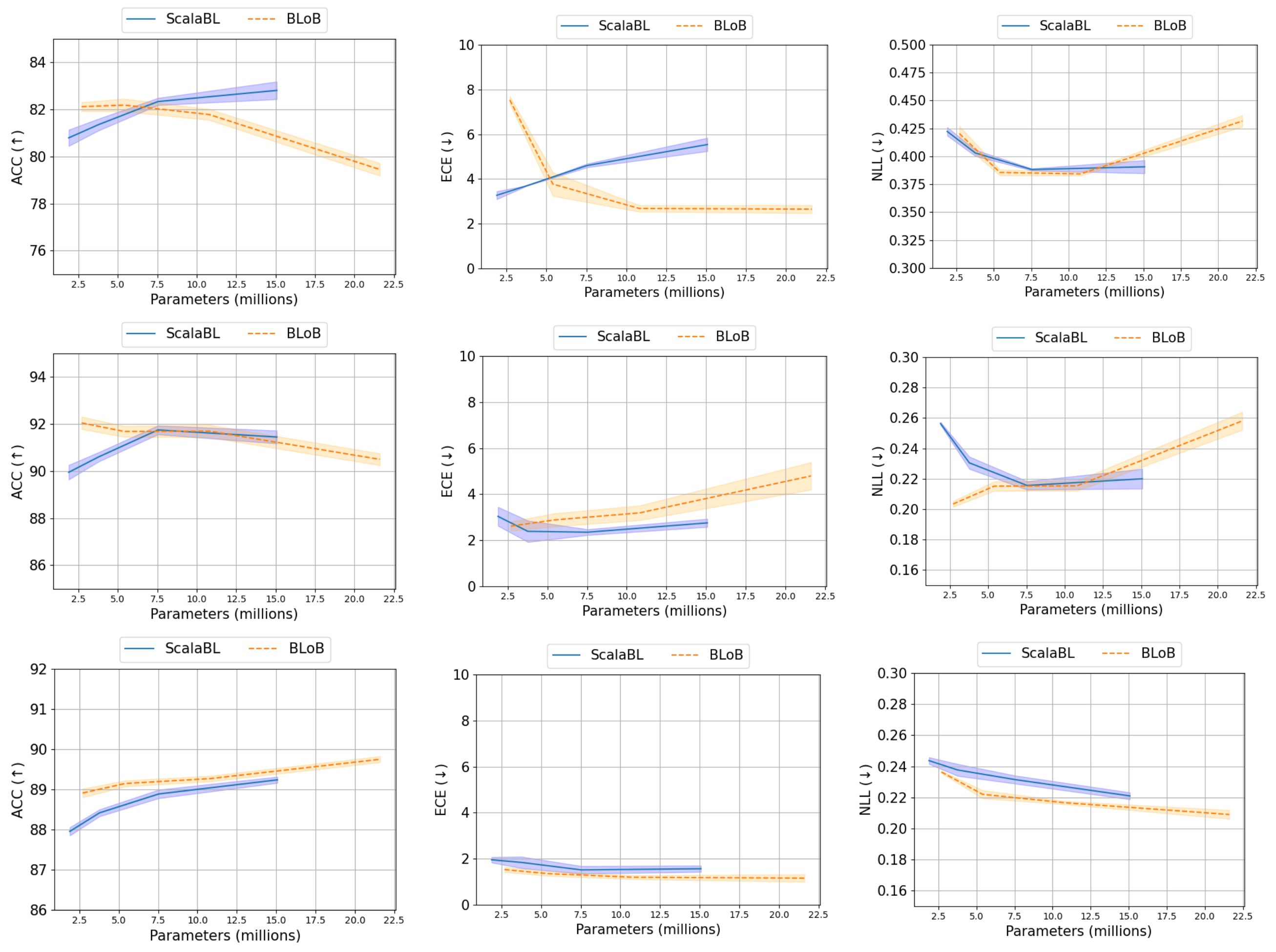

Figure 4: Effect of LoRA rank r. **Top:** Winogrande-Medium (WG-M). **Middle:** OpenBookQA (OBQA). **Bottom:** BoolQ.