Minimal Ranks, Maximum Confidence: Parameter-efficient Uncertainty Quantification for LoRA

Anonymous ACL submission

Abstract

Low-Rank Adaptation (LoRA) enables parameter-efficient fine-tuning of large language models by decomposing weight updates into low-rank matrices, significantly reducing storage and computational overhead. While effective, standard LoRA lacks mechanisms for uncertainty quantification, leading to overconfident and poorly calibrated models. Bayesian variants of LoRA address this limitation, but at the cost of a significantly increased number of trainable parameters, partially offsetting the original efficiency gains. Additionally, these models are harder to train and may suffer from unstable convergence. In this work, we propose a novel parameter-efficient Bayesian LoRA, demonstrating that effective uncertainty quantification can be achieved in very low-dimensional parameter spaces. The proposed method achieves strong performance with improved calibration and generalization while maintaining computational efficiency. Our empirical findings show that, with the appropriate projection of the weight space: (1) uncertainty can be effectively modeled in a low-dimensional space, and (2) weight covariances exhibit low ranks.

1 Introduction

LoRA (Low-Rank Adaptation) (Hu et al., 2021) reduces computational overhead by decomposing the update weights of pre-trained models into low-rank matrices, enabling efficient adaptation to downstream tasks. Minimizing the number of trainable parameters reduces memory and storage requirements, making large-scale model adaptation feasible. Reducing computational overhead speeds up training time and makes adaptation possible in resource-constrained settings.

Unlike pre-trained models, which are relatively well-calibrated (OpenAI, 2023), fine-tuned large models (e.g., LLMs) often become overconfident and poorly calibrated (Jiang et al., 2021; Tian et al., 2023; Xiao et al., 2022; He et al., 2023), especially when trained on limited data. This hinders their usability for applications where uncertainty-aware decisions are performed.

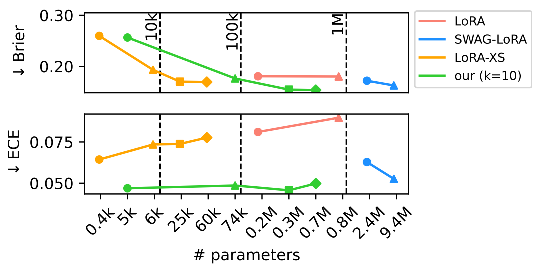
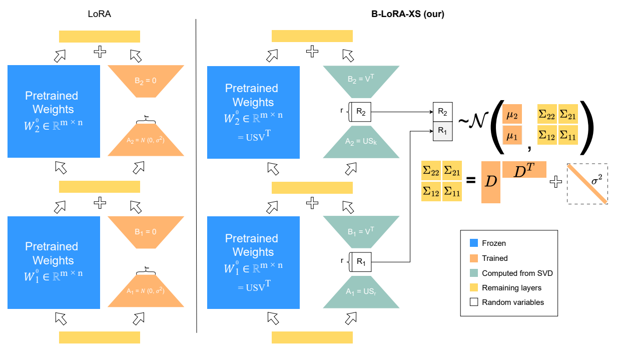


Figure 1: Performance averaged over multiple GLUE datasets (individual results in Fig. 3). Our method achieves superior calibration (ECE) and competitive predictive performance (Brier) while maintaining computational efficiency. For example, at $r = 8$ (▲), we reduce ECE by half with only 1/10th LoRA parameters.

Bayesian treatment is then frequently proposed to address overconfidence in neural networks (Blundell et al., 2015; Kristiadi et al., 2020; Aitchison et al., 2021; Izmailov et al., 2021). Consequently, recently proposed Bayesian variants of LoRA (Onal et al., 2024; Robeyns, 2024; Doan et al., 2025) address the aforementioned challenges by introducing uncertainty estimation directly into the fine-tuning process. During training, these models continuously adjust both the mean and covariance of fine-tuned parameters to achieve better generalization and uncertainty quantification.

Learning the posterior covariance matrix is necessary for modeling epistemic uncertainty. However, its size grows quadratically with the number of parameters, which can easily cancel out the benefits of LoRA, in addition to making learning significantly harder. Using low-rank, Kronecker-factored, or diagonal-only covariances partially alleviates the problem, but as we demonstrate in Sec. 3, this comes at the cost of results quality loss. Furthermore, even at rank = 2, the number of trainable parameters is quadrupled compared to vanilla LoRA. This creates a need for an alternative approach that


LoRA
B-LoRA-XS (our)
Pretrained Weights $W_2^0 \in \mathbb{R}^{m \times n}$
$B_2 = 0$
$A_2 = \mathcal{N}(0, \sigma^2)$
Pretrained Weights $W_1^0 \in \mathbb{R}^{m \times n}$
$B_1 = 0$
$A_1 = \mathcal{N}(0, \sigma^2)$
Pretrained Weights $W_2^0 \in \mathbb{R}^{m \times n} = USV^T$
$B_2 = V^T$
R_2
$A_2 = US_k$
Pretrained Weights $W_1^0 \in \mathbb{R}^{m \times n} = USV^T$
$B_1 = V^T$
R_1
$A_1 = US_r$
r
$\sim \mathcal{N}\left(\begin{matrix}\mu_2 \\ \mu_1\end{matrix}, \begin{matrix}\Sigma_{22} & \Sigma_{21} \\ \Sigma_{12} & \Sigma_{11}\end{matrix}\right)$
$\begin{matrix}\Sigma_{22} & \Sigma_{21} \\ \Sigma_{12} & \Sigma_{11}\end{matrix} = D\,D^T + \sigma^2$
Frozen
Trained
Computed from SVD
Remaining layers
Random variables

	Method	r	k	# Parameters
Standard	LoRA	2	-	0.2M
	LoRA	8	-	0.8M
	LoRA-XS	8	-	6k
	LoRA-XS	25	-	60k
Bayesian	SWAG-LoRA	2	10	2.4M
	SWAG-LoRA	8	10	9.4M
	SWAG-LoRA	8	5	5.5M
	B-LoRA-XS	8	10	74k
	B-LoRA-XS	25	10	0.7M
	B-LoRA-XS	25	5	0.4M

Figure 2: **(left):** Weight-adaptation approaches: LoRA vs. B-LoRA-XS. As indicated by the color coding, some parameters remain frozen (*blue*), others are trained (*orange*) or obtained via SVD (*green*). **(right):** Number of trainable parameters per method. XS variants remain computationally competitive even for ranks as large as $r = 25$.

retains covariance modeling capacity while reducing the number of required parameters.

We propose a method that learns Bayesian posteriors for weights projected onto a low-dimensional manifold, hence maintaining parameter efficiency. The thoughtfully selected projection allows for the effective representation of the covariances between weights through covariances between representations in the lower-dimensional space. In this design, we follow the work of Bałazy et al. (2024), who recently proposed a strategy for finding such projections with SVD. We prove that they are *effective for learning Bayesian models as well*.

Operating in such a reduced parameter space significantly improves the feasibility of Bayesian inference. We show that correlations between weights can be represented very efficiently – unlike in the original weight space, we can use covariance matrices with ranks as low as $k = 2$. Thanks to the low number of parameters, training is also more stable. Finally, the method achieves superior calibration and accuracy at low budgets (e.g., see Fig. 1).

Crucially, the proposed Bayesian learning specifically in the low-rank projected subspace derived from the pre-trained weights is a novel idea and a non-trivial contribution. This approach enables uncertainty-aware fine-tuning in a highly parameter-efficient manner, a challenge not addressed directly in any prior work. While the core components (LoRA-XS projections, Bayesian methods) are known, their synergistic application to learn Bayesian posteriors within such a compressed subspace represents a genuine conceptual innovation, leading to clear empirical benefits in uncertainty quantification with minimal overhead.

In the Appendix, we supplement the results presented in the paper with a discussion of related work and a detailed overview of the experimental setup. We are currently preparing our method's implementation for release.

2 Method

LoRA fine-tunes large pre-trained models by learning low-rank weight updates ΔW instead of training the weights W directly. For a pre-trained parameter matrix $W^0 \in \mathbb{R}^{m \times n}$ that is kept fixed, LoRA learns a rank-r update $\Delta W = AB$, where $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$ have far fewer parameters. The effective weight is then: $W = W^0 + \Delta W = W^0 + AB$, where only A and B are trained. Typically, LoRA is applied jointly at multiple layers, yielding a set of updates $\{\Delta W_l\}$.

Bayesian treatment of a neural network involves finding the posterior $p(\theta \mid \mathcal{D})$ given training data $\mathcal{D}$. By Bayes' theorem: $p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D}|\theta)\, p(\theta)}{p(\mathcal{D})}$, where θ represents the model's parameters (i.e., weights) considered random variables. In particular, for the Bayesian LoRA setting, θ *denotes a set of the learned model updates*, while the remaining frozen weights are hidden inside the model likelihood, given by $p(\mathcal{D} \mid \theta) = \prod_{i \in [\mathcal{D}]} p(y_i|x_i, \theta)$. The learned posterior allows Bayesian model averaging at inference as: $p(y_* \mid x_*, \mathcal{D}) = \int p(y_* \mid x_*, \theta)\, p(\theta \mid \mathcal{D})\, d\theta \approx \frac{1}{S} \sum_{\theta \sim p(\theta|D)} p(y_* \mid x_*, \theta)$, where we use $S = 15$ samples from the posterior.

Bayesian LoRAs obtain the posterior for $\{\Delta W_l\}$ through the learned posterior for $\theta = \cup_l \{A_l \cup B_l\}$, where l indexes the weight updates (layers). The posterior itself is approximated either using a set of particles or a closed-form distribution. Due to its superior performance, we rely on the latter and assume $p(\theta|\mathcal{D}) \approx \mathcal{N}(\mu, \Sigma)$, where μ is the vector of means (of size equal to the number of learned parameters) and Σ is the covariance matrix, whose size grows quadratically with the total number of parameters. Notably, we aim to *model cross-layer interdependencies*, requiring covariance estimation also across weights in different layers $\{l\}$. However, this results in an impractically

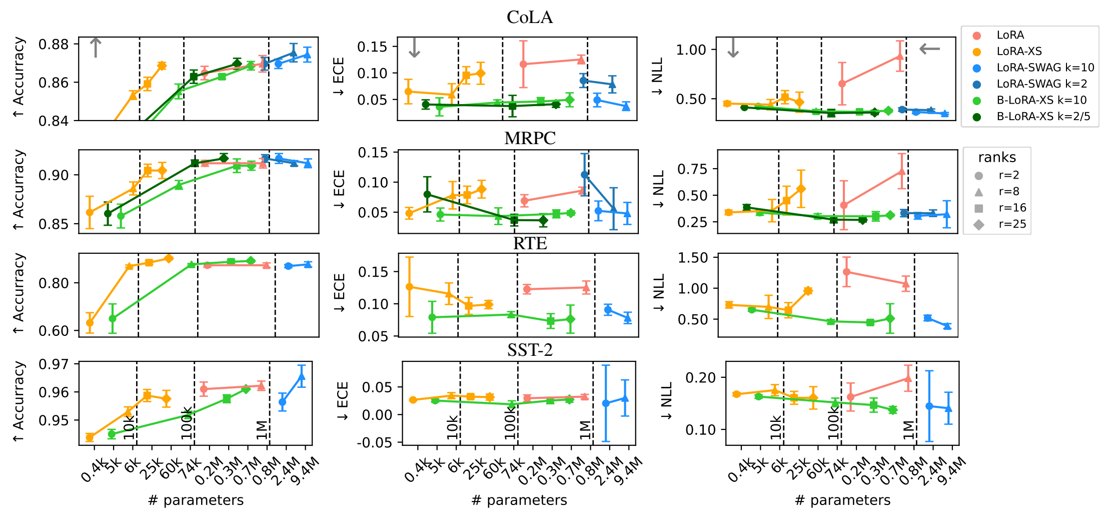


Figure 3: Median±std. accuracy (left), ECE (middle), and NLL (right) on 4 GLUE tasks (rows) vs. total parameter count for several methods and varying ranks r. B-LoRA-XS (our) achieves the accuracy and the calibration of SWAG-LoRA while using significantly fewer parameters than LoRA. See Fig. 1 for averaged results.

large number of parameters. Consequently, we explore methods to reduce this cost by representing distributions $p(\{\Delta W\}|\mathcal{D})$ differently.

In LoRA-XS (Bałazy et al., 2024), the adaptation matrices A and B are initialized using the truncated SVD of the corresponding pre-trained weight matrices W^0. This initialization captures the most informative singular components of the original weights. Under the assumption that the fine-tuned task is similar to the original task, these projections retain the functional properties also for downstream adaptations. LoRA-XS then *freezes* A and B and inserts a small trainable matrix $R \in \mathcal{R}^{r\times r}$ between them, reducing the number of trainable parameters to r^2 ($r^2 \ll (n+m)\cdot r$) per weight matrix. Then, the fine-tuning update is: $h = xW^0 + x\Delta W = xW^0 + xARB$, where $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times n}$ are low-rank matrices obtained from the truncated SVD of W^0, specifically $A = U_r S_r$ and $B = V_r^T$.

B-LoRA-XS, proposed in this paper, leverages the frozen projections A and B for effective and efficient Bayesian learning. The core idea however is not merely applying Bayesian methods to LoRA, but to do so within the extremely compressed parameter space defined by LoRA-XS, making Bayesian inference tractable and highly efficient. This specific idea is new, non-trivial and it required practical validation preceded by design work and tuning. In Sec. 3, we empirically demonstrate that A and B, obtained from the SVD of the pre-trained weights, are not only effective for point-wise fine-tuning but also enable effective uncertainty quantification for $\{\Delta W_l\}$ through modeling covariances for $\{R_l\}$. Although we never compute it explicitly, the covariance matrix for individual ΔW is expressed as $\Sigma_{\Delta W} = (B^T \otimes A)\Sigma_R(B^T \otimes A)^T$, where Σ_R is the (intra-layer) covariance matrix for R and $\otimes$ denotes the Kronecker product.

In practice, we simply learn the joint posterior $p(\theta = \cup_l R_l|\mathcal{D}) \approx \mathcal{N}(\mu, \Sigma)$ for the inner matrices R. The covariance matrix Σ captures both inter-layer and intra-layer dependencies, allowing the model to learn complex relationships. At inference, similar to LoRA, we use samples of R along with the respective projections A and B to obtain h, as realized through samples of ΔW, however without ever computing it explicitly.

The parameters μ and Σ are learned efficiently using **SWAG** (Maddox et al., 2019) (though Variational Inference or the Laplace approximation could also be used). After a burn-in phase (a fixed 10 or 25 epochs) of the gradient-based optimization, the algorithm maintains $\hat{\mu}$ – a running average of θ – along with k vectors of differences $\hat{D}_{last} = \theta_{last} - \hat{\mu}$ for the last k values of θ, and a running average of θ^2. Based on these averages, we estimate the variances $\hat{\sigma}^2$ for individual parameters and approximate the covariance as $\hat{\Sigma} \approx \frac{1}{2}(\hat{D}\cdot\hat{D}^T + diag(\hat{\sigma}^2))$, which constitutes a rank-k approximation to the covariance matrix.

We illustrate B-LoRA-XS method in Fig. 2. Our method uses the total of $|\theta|\cdot(k+2)$ parameters, where $|\theta| = \sum_l r_l^2$.

3 Experiments

Setup: We performed experimental evaluation on four GLUE tasks (Wang et al., 2019) (RTE, MRPC, CoLA, and SST-2) using RoBERTA-large (Liu et al., 2019). We compare our method (B-LoRA-

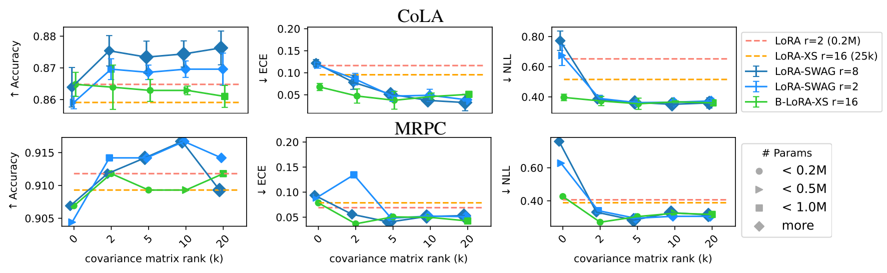


Figure 4: Impact of the posterior covariance matrix rank ($k = 0$ indicates the case with no off-diagonal elements) for CoLA (top) and MRPC (bottom). For brevity, confidence bars ($\pm$ standard deviation) are omitted for MRPC. The colored lines represent non-Bayesian baselines (e.g., standard LoRA or LoRA-XS at a given rank r).

XS) against the standard LoRA, LoRA-XS – a parameter efficient variant, and against SWAG-LoRA (Onal et al., 2024) – a Bayesian variant. For LoRA-XS and B-LoRA-XS we considered ranks $r \in \{2, 8, 16, 25\}$ and for LoRA and SWAG-LoRA due to limited budget we were able to test $r \in \{2, 8\}$. The number of parameters (a *proxy for storage and computation costs*) as a function of ranks r and k is summarized in Fig. 2. We report accuracy (higher is better), ECE and NLL (lower is better) of median$\pm$std across 5 runs.

Performance analysis: Fig. 3 compares accuracy, Expected Calibration Error (ECE), and Negative Log-Likelihood (NLL) against total parameter count across 4 datasets. Our main claim is that B-LoRA-XS improves overall model performance, with a particular focus on calibration metrics. Indeed, Figure 3 (middle and right) demonstrates that B-LoRA-XS consistently yields lower ECE and NLL compared to standard LoRA across all parameter scales. Regarding accuracy (Figure 3: left), while standard LoRA shows marginally better results for a few configurations at moderate parameter scales, the majority of configurations show B-LoRA-XS matching or exceeding the accuracy of standard LoRA. More importantly, in no setting does standard LoRA significantly outperform B-LoRA-XS in terms of calibration, which is a primary focus of our work. Bayesian variants, including B-LoRA-XS and SWAG-LoRA, generally outperform their non-Bayesian counterparts in ECE and NLL. However, our model achieves these strong calibration results with 5–15 times fewer parameters than SWAG-LoRA. Moreover, while SWAG-LoRA sometimes performs well, its results vary significantly between runs. In contrast, B-LoRA-XS exhibits stable and consistent convergence. Finally, as results for MRPC and CoLA suggest, its performance remains robust across different values of k, whereas SWAG-LoRA's ECE deteriorates significantly at $k = 2$.

Covariance matrix rank analysis: Figure 4 compares the sensitivity of the Bayesian LoRA variants to changes in covariance matrix rank k. Markers indicate model sizes (e.g., SWAG-LoRA $\gg$ B-LoRA-XS). As expected, SWAG-LoRA deteriorates proportionally as rank decreases. On the other hand, B-LoRA-XS maintains its performance across a wide range of k. Significant degradation occurs only when off-diagonal covariance values are entirely ignored (i.e., at $k = 0$). Notably, B-LoRA-XS achieves the best calibration at low ranks, particularly at $k = 2$ or $k = 5$. This demonstrates that the SVD-based projection effectively disentangles parameters, enabling low-dimensional uncertainty modeling.

Data size reduction analysis: Due to space constraints, the experiment was moved to Section F.

4 Conclusion

B-LoRA-XS addresses the lack of uncertainty quantification in LoRA fine-tuning while maintaining parameter efficiency. It utilizes truncated SVD to project model updates into a lower-dimensional space and leverages the Bayesian framework to enhance uncertainty estimation.

Our method's primary strength lies in its calibration capabilities; it consistently achieves *lower expected calibration error and negative log-likelihood* compared to standard LoRA and LoRA-XS across various parameter scales. While standard LoRA may exhibit marginally better accuracy in a few specific configurations, B-LoRA-XS generally matches or exceeds its accuracy in most settings, and critically, always provides superior or equal calibration. Compared to the Bayesian LoRA baseline, B-LoRA-XS matches or surpasses its accuracy and calibration performance while *using significantly fewer parameters, exhibiting greater training stability, and relying on simpler, lower-rank covariance representations.*

Limitations

While B-LoRA-XS demonstrates promising results in parameter-efficient uncertainty quantification, several limitations should be acknowledged. First, the effectiveness of B-LoRA-XS inherently depends on the quality of the initial SVD projection derived from pre-trained weights (as in LoRA-XS). If the principal components of the pre-trained model are not well-aligned with the requirements of a significantly different downstream task, the performance might be suboptimal. Second, our method employs SWAG with a low-rank approximation for the covariance matrix. While efficient, this is one specific approach to approximate Bayesian inference. Other techniques (e.g., more sophisticated variational inference methods or different posterior approximations) might yield different trade-offs between performance, calibration, and computational cost, and were not explored in this work. Third, although B-LoRA-XS significantly reduces the number of trainable parameters for Bayesian adaptation, the inference process still requires multiple forward passes for sampling, which increases computational cost compared to non-Bayesian LoRA or LoRA-XS. This trade-off between improved uncertainty and inference overhead is an important consideration for deployment. Four our empirical validation is conducted on GLUE classification tasks using RoBERTa-Large. The generalizability of B-LoRA-XS's benefits to other model architectures, much larger model scales, or different task types (such as text generation or more complex reasoning tasks) warrants further investigation. Finally, the optimal choice of LoRA rank r and SWAG covariance rank k might require careful tuning for different datasets and models, potentially adding to the practical overhead of applying the method effectively.

References

Laurence Aitchison, Adam Yang, and Sebastian W Ober. 2021. Deep kernel processes. In *International Conference on Machine Learning*, pages 130–140. PMLR.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2024. Lora-xs: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Bao Gia Doan, Afshar Shamsi, Xiao-Yu Guo, Arash Mohammadi, Hamid Alinejad-Rokny, Dino Sejdinovic, Damien Teney, Damith C. Ranasinghe, and Ehsan Abbasnejad. 2025. Bayesian low-rank learning (bella): A practical approach to bayesian neural networks. *Preprint*, arXiv:2407.20891.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Guande He, Jianfei Chen, and Jun Zhu. 2023. Preserving pre-trained features helps calibrate fine-tuned language models. In *ICLR*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. 2021. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. 2020. Being Bayesian, even just a bit, fixes overconfidence in relu networks. In *ICML*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Cristian Meo, Ksenia Sycheva, Anirudh Goyal, and Justin Dauwels. 2024. Bayesian-loRA: LoRA based parameter efficient fine-tuning using optimal quantization levels and rank values trough differentiable bayesian gates. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*.

Emre Onal, Klemens Flöge, Emma Caldwell, Arsen Sheverdin, and Vincent Fortuin. 2024. Gaussian stochastic weight averaging for bayesian low-rank adaptation of large language models. In *Sixth Symposium on Advances in Approximate Bayesian Inference - Non Archival Track*.

OpenAI. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Maxime Robeyns. 2024. Bayesian low-rank adaptation for large language models. In *ICLR*.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2022. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. In *EMNLP*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

A Related Work

PEFT: As large language models continue to grow, parameter-efficient fine-tuning (PEFT) has become a popular approach to reducing computational and storage costs. Among various methods (Houlsby et al., 2019; Guo et al., 2021; Li and Liang, 2021; Lester et al., 2021), LoRA (Hu et al., 2021) has emerged as one of the most widely used. Building on its success, several approaches have been proposed to enhance different aspects of PEFT (Kopiczko et al., 2023; Zhang et al., 2023; Dettmers et al., 2024). One such method, LoRA-XS (Bałazy et al., 2024), further optimizes parameter efficiency by enabling flexible control over the number of trainable parameters per adaptation module. B-LoRA-XS reuses the idea of SVD-based projections to reduce the parameter space dimensionality.

Bayesian LoRAs: Standard LoRA (Hu et al., 2021) does not account for uncertainty, making fine-tuned models susceptible to miscalibration. Then, Bayesian LoRA approaches integrate Bayesian inference techniques into LoRA to improve uncertainty estimation and generalization.

Several Bayesian LoRA methods have been proposed, each employing different Bayesian techniques to address these challenges. SWAG-LoRA (Onal et al., 2024) combines Stochastic Weight Averaging-Gaussian (SWAG) with LoRA to enable approximate Bayesian inference, significantly improving model calibration and reducing overconfidence. Laplace-LoRA (Robeyns, 2024) applies a Laplace approximation to the posterior over LoRA parameters. Bella (Doan et al., 2025) introduces an approach that reduces the cost of Bayesian deep ensembles by applying multiple low-rank perturbations to a pre-trained model. BLoB (Bayesian Low-Rank Adaptation by Backpropagation) (Wang et al., 2024) jointly learns both the mean and covariance of model parameters throughout the fine-tuning process using Variational Inference. B-LoRA (Meo et al., 2024) introduces a Bayesian perspective to both quantization and rank selection by using a prior distribution over these hyperparameters, optimizing model efficiency and reducing bit operations.

The key challenge lies in balancing uncertainty modeling with parameter efficiency, as Bayesian inference typically increases both the number of trainable parameters and computational cost. Despite their advantages, Bayesian LoRA methods face challenges related to increased parameter count and computational cost. One major issue is the higher storage and memory requirements, as Bayesian methods often require additional parameters to model uncertainty, particularly those involving covariance estimation, such as SWAG-LoRA. Scalability remains a concern for methods that explicitly model uncertainty across a large number of parameters.

B Scientific Artifacts Licenses

Listed below are the licenses for the scientific artifacts used in this research. For complete information, please use the links below and refer to the original sources.

Scientific Artifacts: RoBERTa-large (MIT), MRPC (Unknown), RTE (Unknown), COLA (Unknown), SST-2 (Unknown), HuggingFace Transformers Library (Apache-2.0), SWAG-LoRa repository[1] (MIT), LoRa-XS repository[2] (Unknown).

C Model Size And Budget

- RoBERTA-large: 355M parameters
- GPUs: RTX4090 and V100-SXM2-32GB, each run was performed on a single GPU
- GPU total time: $\approx$ 63 days

D Statistics For Data

We followed the original GLUE train-validation split

- MRPC - train: 3'668, val: 408
- RTE - train: 2'490, val: 277
- CoLa - train: 8'551, val: 1043
- SST2 - train: 67'349, val: 872

E Experimental Setup Details

The study was conducted on a subset of the GLUE benchmark (Wang et al., 2019), specifically on RTE, MRPC, CoLA, and SST-2 tasks (with the original train-validation split), using RoBERTa-large (Liu et al., 2019) checkpoints from the HuggingFace Transformers library (Wolf et al., 2020). For the RTE and MRPC tasks, we followed LoRA-XS and initialized LoRA-XS modules with

[1] `https://github.com/fortuinlab/swag-lora`

[2] `https://github.com/MohammadrezaBanaei/LoRA-XS`

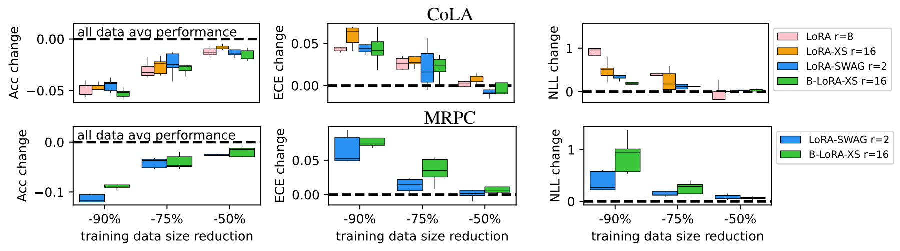


Figure 5: Accuracy, ECE and NLL change as the training set is progressively reduced (e.g. -90% means using only 10% of the data for training). The dashed line marks the model's performance when trained on the full dataset.

weights fine-tuned on the MNLI task. We integrated B-LoRA-XS/LoRA-XS modules into the Query, Value, Attention Output, and Output Fully Connected weight matrices in all transformer layers (Vaswani et al., 2017), whereas due to budget limits, standard LoRA and SWAG-LoRA modules were added only to the Query and Value matrices. Note, this is sufficient for SWAG-LoRA to achieve its best performance.

For each dataset, for the burn-in stage of training, we adopted hyperparameters from the LoRA-XS paper. These include: learning rate, batch size, AdamW optimizer (Loshchilov and Hutter, 2019), linear scheduler with warm-up, dropout, and the LoRA scaling factor α. For standard LoRA we followed the same setup, except for the learning rate, which was determined through grid search. Similarly, the SWAG starting epoch (e.g. 10 or 25) was selected through grid search. Based on the findings from SWAG-LoRA, we used a constant learning rate scheduler (SWALR) with warm-up. The SWAG learning rate was set to the maximum learning rate from the first (burn-in) phase of training. Unless stated otherwise, we used a low-rank covariance matrix approximation with the rank $k = 10$. In all our experiments, SWAG estimation was applied exclusively to the LoRA modules, and SWAG predictions were consistently obtained with $S = 15$ model samples.

F Additional Experimental Results

Data size reduction analysis: Figure 5 compares how accuracy, ECE, and NLL degrade when training data is subsampled. All methods predictably lose accuracy as data size decreases, with little difference between the various LoRA-based approaches. We conclude that Bayesian learning does not improve robustness in this case. However, we note variations across datasets in terms of accuracy. For example, in MRPC, the decline is more pronounced, likely due to the dataset smaller size.

G Acknowledgments

We acknowledge the use of ChatGPT for grammar checking and generation of the initial version of the plotting code.