Tied-LoRA: Enhancing parameter efficiency of LoRA with Weight Tying

Anonymous ACL submission

Abstract

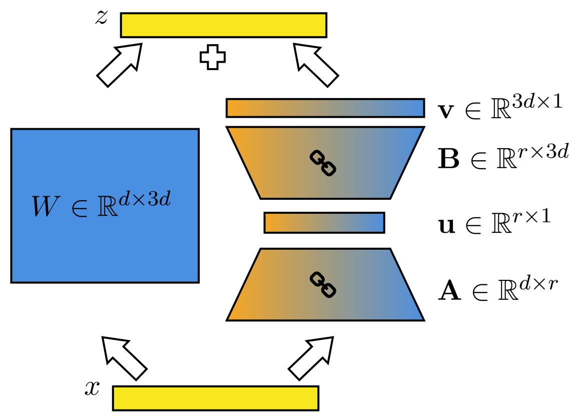
We introduce Tied-LoRA, a novel paradigm leveraging weight tying and selective training to enhance the parameter efficiency of Low-rank Adaptation (LoRA). Our exploration encompasses all plausible combinations of parameter training and freezing, coupled with weight tying, aimed at identifying the optimal trade-off between performance and the count of trainable parameters. Across 5 diverse tasks and two foundational language models, our experiments provide comprehensive insights into the inherent trade-offs between efficiency and performance.

Our findings reveal a specific Tied-LoRA configuration that distinguishes itself by showcasing comparable performance to LoRA across multiple tasks while utilizing only a fraction of the parameters employed by the standard LoRA method, particularly at elevated ranks. This underscores the efficacy of Tied-LoRA in achieving impressive results with significantly reduced model complexity.

1 Introduction

Large language models (LLMs) play a crucial role in various Natural Language Processing (NLP) applications due to their proficiency. A significant factor driving their widespread adoption is the ability to fine-tune pretrained LLMs efficiently for specific downstream tasks. This fine-tuning process allows the creation of specialized language models that excel in specific domains and tasks. Despite dealing with smaller training data compared to pretraining, the computational demand for during fine-tuning remains high, especially for large models with billions of parameters.

Moreover, for LLM service providers, it is often necessary to cater to diverse requirements by maintaining distinct customizations for each combination of user and task in the service. For instance, consider a scenario where a language model is employed to assist users in generating content



z
$\mathbf{v} \in \mathbb{R}^{3d\times 1}$
$W \in \mathbb{R}^{d\times 3d}$
$\mathbf{B} \in \mathbb{R}^{r\times 3d}$
$\mathbf{u} \in \mathbb{R}^{r\times 1}$
$\mathbf{A} \in \mathbb{R}^{d\times r}$
x

Figure 1: Schematic of our Tied-Lora paradigm, the main low-rank matrices A and B are tied across (indicated by the 🔗 symbol) all the layers of the base language model. We use the gradient shading to indicate that these parameters can either be trained or frozen.

for social media. User X may have preferences for formal language and professional tone, while another user, Y, might prefer a more casual and conversational style. Additionally, each user may have different tasks, such as composing business emails, translating documents, creating social media captions or drafting blog posts. To serve these varied preferences and tasks simultaneously, the language model needs to be finely tuned for each specific combination of user (X or Y) and task (email composition or translation).

As the number of users and tasks per user increases, so does the complexity and cost associated with customization. Managing and storing the various combinations of customizations for each user-task pair can introduce additional expenses, especially after the initial training phase. The storage and retrieval of these customized models, each tailored to specific user preferences and tasks, contribute to ongoing operational costs. Therefore, in addition to efficient utilization of customizable parameters during training, careful consideration must be given to the post-training phase, where the cost of saving and accessing these combinations becomes a significant factor in the overall resource management strategy. This holistic approach is crucial for maintaining optimal performance across diverse user-task combinations while keeping both computational and operational costs in check.

In light of these challenges, developing effective customization methods that not only enhance model performance but also reduce the number of training parameters becomes crucial. Parameter-efficient fine-tuning (PEFT) emerges as a valuable approach in this context. PEFT involves refining pretrained models with minimal parameter updates, enabling the creation of specialized models that excel in specific domains and tasks. This streamlined customization process not only optimizes parameter utilization during training but also mitigates the costs associated with managing, storing, and serving diverse customizations post-training.

Low-rank Adaptation (LoRA) (Hu et al., 2021) has emerged as a popular (PEFT) method because of its straightforward implementation and the ability to merge LoRA weights into the base model. However, despite its advantages, LoRA training can still be expensive, especially as the base models become increasingly larger. While prior work has attempted to make LoRA more parameter efficient, they concentrated on appropriate low-rank selection. However, we introduce a novel approach, Instead of controlling the number of parameters by the rank, we employ simple weight tying coupled with selective training. By integrating these two core ideas, we propose a range of Tied-LoRA configurations and study the performance of each configuration on five diverse customization tasks.

Low-rank Adaptation (LoRA) method (Hu et al., 2021), stands out as a popular and efficient parameter-efficient fine-tuning (PEFT) approach, offering a straightforward implementation and the ability to integrate LoRA weights into the base model post-training. Despite its advantages, the expense of LoRA training becomes more pronounced, particularly with the growing size of base language models. While previous efforts focused on enhancing LoRA's parameter efficiency through careful low-rank selection, we introduce an alternative approach. In contrast to controlling parameter count through rank, our method incorporates simple weight tying alongside selective training. This novel combination forms the basis for a range of Tied-LoRAconfigurations, each evaluated for performance across five diverse customization tasks. Through this approach, we aim to push the boundaries of parameter-efficient fine-tuning, making advancements in both effectiveness and simplicity. Our contributions are threefold:

1. We propose a range of Tied-LoRA configurations that use simple weight tying in LoRA along with selective training to boost the parameter efficiency of LoRA.

2. We study this spectrum of possible Tied-LoRA configurations on diverse tasks that resemble real-world customization problems.

3. Based on the results of our study, we propose the specific $\mathrm{TL}_6(\mathbf{v}\mathbf{B}_{⚭}\mathbf{u}\mathbf{A}_{⚭})$ configuration as the best option for maintaining performance while reducing parameters. This configuration is within $1 - 2\%$ of LoRA in terms of performance and in one case beats LoRA while only using 12.5% of the number of parameters.

2 Method

In this section, we introduce Tied-LoRA, a paradigm for parameter-efficient fine-tuning of large language models through low-rank weight-update approximations, weight-tying and selective training. Our framework offers a range of "LoRA-like" configurations through a series of design choices over selective parameter training and weight tying, including some of the existing PEFT methodologies available in the literature. Specifically, we use weight tying alongside pairs of projection matrices and scaling vectors that can be selectively either trained or frozen. As the low-rank computation path does not introduce any non-linearity, all Tied-LoRAconfigurations can be merged into the base model weights to preventing additional latency during inference.

Table 1 provides an overview of the scenarios we study. We refer to each configuration in our study with TL (Tied-LoRA) followed by a subscript index (e.g., TL_1). Additionally, we also include the template of possible training parameters ($\mathbf{v}, \mathbf{B}, \mathbf{u}$ and $\mathbf{A}$, discussed in section 2.1). For Tied-LoRA, the low-rank projection matrices $\mathbf{A}$ and $\mathbf{B}$ are *tied* across all the layers of the base model which we indicate using the subscript $_{⚭}$. We also indicate if a parameter is frozen by blue font and a trainable parameter with regular font. Thus, traditional LoRA can be expressed as ${\color{blue}\mathbf{v}}\mathbf{B}{\color{blue}\mathbf{u}}\mathbf{A}$ and VeRA (Kopiczko et al., 2023) can be expressed as $\mathbf{v}{\color{blue}\mathbf{B}_{⚭}}\mathbf{u}{\color{blue}\mathbf{A}_{⚭}}$.

2.1 Formulation

The overall structure of the tied LoRA framework can be seen in Figure 1. Note that the original LoRA (Hu et al., 2021) uses a dedicated pair of low-rank projections for each of the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ matrices. However, in our formulation, W is a $d \times 3d$ matrix that jointly projects $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ attention matrices, where d is the hidden size of the base language model. Therefore, our down projection $\mathbf{A}$ is a $d \times r$ shaped matrix and up projection matrix $\mathbf{B}$ has shape $r \times 3d$, where r is the low-rank bottleneck dimension. Essentially, the down projection $\mathbf{A}$ is *shared* by $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$, leading to fewer trainable parameters ($4dr$) than the original LoRA ($6dr$).

For a linear layer with a frozen pretrained weight matrix $\mathbf{W}$, we define the layer output as

$$z = \mathbf{W}x + \Delta\mathbf{W}x \approx \mathbf{W}x + \frac{\alpha}{r}\Lambda_v \mathbf{B} \Lambda_u \mathbf{A} x, \quad (1)$$

where $\Delta\mathbf{W}$ is the full-rank update matrix, α is a scaling factor, $\mathbf{A}$ and $\mathbf{B}$ are low-rank projection matrices, and Λ_u and Λ_v are diagonal matrices with diagonal elements given by u and v, respectively. Herein, $\Lambda_v \mathbf{B} \Lambda_u \mathbf{A} x$ is the low-rank approximation to the parameter update matrix $\Delta\mathbf{W}$. Unlike the original LoRA, where α is a hyper-parameter that can be manually set, we simply set $\alpha = r$, effectively removing its scaling effect.

Equation 1 is a generalized formulation for methods that utilize low-rank approximations to estimate parameter updates. Particular settings of parameter updates and weight tying reduces this equation to some of the existing formulations in the literature. Setting and freezing $\Lambda_u = \Lambda_v = I$ and untying $\mathbf{A}$ and $\mathbf{B}$ results in LoRA:

$$z = \mathbf{W}x + \mathbf{B}\mathbf{A}x. \quad (2)$$

Similarly, randomly initializing $\mathbf{A}$ and $\mathbf{B}$ matrices and tying them across all layer leads the the VeRA formulation (Kopiczko et al., 2023):

$$z = \mathbf{W}x + \Lambda_v \mathbf{B} \Lambda_u \mathbf{A} x, \quad (3)$$

2.2 Weight Tying

The third column of Table 1 presents representations for number of trainable parameters each Tied-Lora configuration requires. As is apparent from the table, weight tying is a critical ingredient of our proposed approach which drastically reduces

Method	Parameters	Initialization
LoRA ($\mathbf{vBuA}$)	$4Ldr$	$A \sim \mathcal{N}, B = 0, u, v = 1$
Vera ($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	$L(r + 3d)$	$A, B \sim \mathcal{N}, u = 1, v = 0$
TL1($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	dr	$A, B \sim \mathcal{N}, u, v = 1$
TL2($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	$dr + L(r + 3d)$	$A, B \sim \mathcal{N}, u = 1, v = 0$
TL3($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	$3dr$	$A, B \sim \mathcal{N}, u, v = 1$
TL4($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	$(L + 3d)r$	$A, B \sim \mathcal{N}, v, u = 1$
TL5($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	$4dr$	$A \sim \mathcal{N}, B = 0, u, v = 1$
TL6($\mathbf{v}\mathbf{B}_{🔗}\mathbf{u}\mathbf{A}_{🔗}$)	$4dr + L(r + 3d)$	$A, B \sim \mathcal{N}, u = 1, v = 0$

Table 1: Tied-LoRAconfigurations included in our study. The first column shows acronyms used to identify each Tied-LoRAconfiguration (i.e., method). Symbols with subscript 🔗 indicate that it is shared across all layers and the color blue indicates that the parameter is frozen. Formulas for the number of trainable parameters in each configuration as a function of number of layers L, hidden size d, and low-rank r are also provided.

the number of trainable parameters. For example, LoRA ($\mathbf{vBuA}$) training using the 7B LLaMA-2 (Touvron et al., 2023) language model with a typical low rank setting of 8 requires $\sim$ 4.2M trainable parameters. By merely introducing weight tying across the 32 layers of this model reduces the trainable parameters to $\sim$ 131K, which is a 96.875% reduction. In comparison, the Vera method results in a reduction of 90.6%.

2.3 Selective Training

Through the flexible framework that equation 1 offers, we are given the opportunity to investigate a range training configurations. By selectively updating the components A, B, u, and v during the training process, we can generate a variety of methodological variations. These variations not only exhibit differences in parameter count, but they also demonstrate distinct capabilities across a variety of tasks. This exploration allows us to investigate the intriguing regime of extremely low-parameter and low-rank PEFT models. This is a key step towards the customization of models, enabling them to excel at specific tasks while maintaining a minimal parameter count. Our ultimate goal here is to harness the power of this methodology to create highly efficient, task-specific models that achieve high performance with reduced complexity.

3 Experiments

We now turn to evaluating the different configurations possible within our Tied-LoRAparadigm. While LoRA ($\mathbf{vBuA}$) and PEFT methods can be used to train models for general instruction following (Sun et al., 2023; Lermen et al., 2023; Sun et al., 2023), we focus our evaluations in a “task customization” perspective, where each model is trained on a specific task and is evaluated on a test set from the same task.

3.1 Tasks & Datasets

To evaluate the performance of each Tied-LoRAconfiguration across diverse data settings, we utilized the following types of tasks:

Extractive QA is a common task where the model is expected to “read” some relevant text (the context) and answer questions. The answers are usually exact sub-strings from the provided context. We use SQuADv1 dataset (Rajpurkar et al., 2016) in our experiments. Since the official test split of this dataset does not contain ground-truth answers, we use the validation set as our test set. We create a validation set comprising of a random sample of 4800 examples extracted from the training set.

Summarization is a central problem in NLP and several variations of summarization datasets have been proposed. We employ the DialogSum dataset (Chen et al., 2021) to study our models’ performance on this task. DialogSum includes summaries of real-word conversations on a diverse set of topics and scenarios. This dataset was an attractive option as the length of the conversations and summarizes are within the context lengths (4096 tokens) of the base language models.

Commonsense Natural Language Inference (NLI) is a task designed to probe the ability of language models to apply “commonsense reasoning” to choose a possible ending for a given situation described in natural language. These tasks are typically trivial for humans but language models can still struggle. We use the HellaSwag dataset (Zellers et al., 2019) to study the performance of our proposed models on this type of task. As HellaSwag contains multiple-choice questions, it can be viewed as a classification problem.

Translation Machine translation is a natural language generation task which is widely used in research and industry. Translation is inherently multilingual and thus offers a challenging domain to study our Tied-LoRAparadigm. There are several large scale translation datasets but we focus on a moderately sized IWSLT 2017 German-to-English spoken language translation dataset (Cettolo et al., 2017). With over $206k$ training examples this is the largest dataset we study.

Mathematical Reasoning is a challenging domain where large language models still lag behind human performance. Using PEFT methods on such tasks further amplifies these challenges as there are very few trainable parameters. In our experiments, we use the GSM8K benchmark (Cobbe et al., 2021) which contains 8.5K high-quality, grade-school level math word problems. Each example in the GSM8K benchmark contains a question and an answer. The answers are provided with natural language solutions which contain explanations of each step used to obtain the final answer. The final numerical answer is demarcated from the rest of the natural language solution. We evaluate our models by comparing these final numerical answers.

3.2 Base Language Models

Although PEFT enables the base language model to perform new tasks, the final performance heavily depends on the inherent abilities learned during pretraining. This necessitates investigating the performance of Tied-LoRAon multiple base models with different inherent capabilities. Therefore, we use a relatively small two billion parameter, GPT-2B-001 model[1] released by NVIDIA and the moderately large 7B LLaMA 2 model (Touvron et al., 2023) released by Meta.

In addition to the size differences, these models also differ in the amount of pretraining data used. The GPT-2B-001 model was trained on 1.1 trillion tokens of text from publicly available multilingual text spanning 53 languages. The LLaMA2 7B model was trained on 2 trillion tokens of predominately English text. Both models are autoregressive language models with a context size of 4096 tokens.

3.3 Implementation Details

We use the open-source NeMo Framework to implement all the algorithms presented in this paper. Our implementation is publicly available through the NeMo GitHub repository.[2] We set max training steps to $2k$, but training was terminated sooner using early stopping with a patience of 10 to prevent over-fitting. We trained all configurations using AdamW optimizer (Loshchilov and Hutter, 2017) with a weight decay of 0.01 and a cosine learning rate schedule with 50 warm-up steps.

For each Tied-Lora method we tried two learning rates, a high rate of 1^{-4} and a low learning rate

[1] `https://huggingface.co/nvidia/GPT-2B-001`
[2] `anonymized`

Base Model	Method	Dialogsum			GSM8K			HellaSwag			IWSLT 2017			Squad		
		RougeL	r	P%	EM	r	P%	Acc.	r	P%	BLEU	r	P%	EM	r	P%
LLaMA2 7B	LoRA ($\mathbf{vBuA}$)	40.76	8	100	32.75	64	100	91.97	16	100	41.30	8	100	88.52	2	100
	Vera ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	38.77	8	9.4	27.22	64	1.2	89.91	16	4.7	40.22	8	9.4	87.69	2	37.5
	TL$_1$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	38.73	8	0.8	27.07	64	0.8	90.03	16	0.8	40.34	8	0.8	87.72	2	0.8
	TL$_2$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	38.69	8	10.2	27.07	64	2.0	90.11	16	5.5	40.35	8	10.2	87.67	2	38.3
	TL$_3$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	40.20	8	2.3	17.74	64	2.3	89.38	16	2.3	39.93	8	2.3	87.34	2	2.3
	TL$_4$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	39.46	8	2.3	21.00	64	2.3	89.46	16	2.3	40.34	8	2.3	87.06	2	2.3
	TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	**40.62**	8	3.1	30.33	64	3.1	**91.75**	16	3.1	40.01	8	3.1	87.11	2	3.1
	TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	39.24	8	12.5	**31.77**	64	4.3	91.15	16	7.8	**41.33**	8	12.5	**87.97**	2	40.6
	Vera ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	40.07	64	9.4	29.11	16	1.2	90.47	2	4.7	40.41	16	9.4	87.69	2	37.5
	TL$_1$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	39.74	16	1.6	29.95	16	0.2	90.52	4	0.2	40.52	64	6.3	87.72	2	0.8
	TL$_2$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	39.81	64	15.7	28.73	16	1.4	90.32	2	4.8	40.50	128	22.0	87.67	2	38.2
	TL$_3$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	40.20	8	2.3	24.34	4	0.1	90.27	8	1.2	40.48	16	4.7	87.62	8	9.4
	TL$_4$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	40.17	16	4.7	25.70	8	0.3	90.18	4	0.6	40.65	16	4.7	87.72	4	4.7
	TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	**40.62**	8	3.1	30.33	64	3.1	91.75	16	3.1	**41.37**	16	6.3	88.22	4	6.3
	TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	39.71	16	15.6	**31.77**	64	4.3	**91.90**	64	17.2	**41.37**	32	21.9	**88.49**	4	43.8
GPT-2B-001	LoRA ($\mathbf{vBuA}$)	38.59	4	100	12.28	64	100	85.64	64	100	40.19	128	100	83.58	32	100
	Vera ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.02	4	18.8	6.97	64	1.2	75.94	64	1.2	38.20	128	0.6	79.43	32	2.3
	TL$_1$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.11	4	1.0	8.26	64	1.0	76.32	64	1.0	38.12	128	1.0	79.26	32	1.0
	TL$_2$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.00	4	19.8	8.11	64	2.2	77.02	64	2.2	38.17	128	1.6	79.50	32	3.4
	TL$_3$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	36.50	4	3.1	5.69	64	3.1	25.05	64	3.1	36.46	128	3.1	76.96	32	3.1
	TL$_4$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	36.82	4	3.1	6.82	64	3.1	25.05	64	3.1	32.98	128	3.1	77.47	32	3.1
	TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.17	4	4.2	8.34	64	4.2	82.25	64	4.2	38.58	128	4.2	81.43	32	4.2
	TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	**37.63**	4	22.9	**9.78**	64	5.3	**85.02**	64	5.3	**39.74**	128	4.8	**83.02**	32	6.5
	Vera ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.28	8	18.8	8.26	2	1.2	83.41	2	1.2	39.15	2	0.6	81.77	2	2.3
	TL$_1$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.22	8	2.1	9.55	4	0.1	83.54	4	0.1	39.09	2	0.1	82.20	2	0.1
	TL$_2$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.29	8	20.1	9.40	4	1.2	83.64	2	1.2	39.11	2	0.6	82.41	4	2.5
	TL$_3$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.18	16	12.5	6.97	8	0.4	80.66	4	0.2	38.25	4	0.1	80.96	4	0.4
	TL$_4$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	36.88	32	25.1	7.20	32	1.6	80.51	4	0.2	38.30	8	0.2	81.03	8	0.8
	TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	37.55	8	8.3	9.40	128	8.3	83.71	32	2.1	39.20	64	2.1	82.74	16	2.1
	TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)	**37.81**	32	52.2	**10.31**	16	2.2	**85.13**	32	3.3	**39.74**	128	4.8	**83.56**	64	10.8

Table 2: The results entire spectrum of Tied-LoRA configurations on five tasks using LLaMA2 7B base model and the GPT-2B-001 base model. For each base model section, the first row shows the best LoRA ($\mathbf{vBuA}$) scores on each task along with rank r at which the best score was achieved.

of 1^{-5}. While the "typical" range of the low-rank dimension r is $4-16$ we find that some complex tasks benefit from higher r so we trained all our models with a wide range of $r \in \{2, 4, 8, \ldots, 128\}$. Each task was trained with a global batch size of 256 and a validation check interval of 30 steps. The only exception was the IWSLT translation dataset for which we set global batch size and validation check interval of 1024 and 60 respectively. No extensive hyper-parameter search was conducted. During inference, we used greedy-decoding to generate the models' predictions with a limit of 500 tokens.

4 Results

Table 2 provides a detailed comparison of various Tied-LoRAconfigurations across our 5 tasks for the LLaMA2 7B and GPT-2B-001 base models. For each task we report the metric used, such as RougeL (Lin and Och, 2004), Exact Match (EM), Accuracy and BLEU (Papineni et al., 2002), and the rank r used. For each model, the table is segmented into two sections: The first section compares the performance of all Tied-LoRAconfigurations at the *same rank* where LoRA ($\mathbf{vBuA}$) achieved its optimum score. The second section shows the best performance of achieved by each configuration. In addition to metric scores and rank (r) we also report parameter usage percentage (P%) as a comparison to the parameter count of the best-performing Lora configuration. This offers a direct measure of efficiency, showing how each model, especially TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) , leverages a smaller percentage of parameters compared to the LoRA ($\mathbf{vBuA}$) for achieving its results.

We can immediately see that LoRA ($\mathbf{vBuA}$) is the best performing model for both the 2B and 7B base language models on most tasks. This is hardly surprising as it is the most expensive method with respect to trainable parameters. The TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) configuration demonstrates the best overall performance among all Tied-LoRA configurations for both model sizes and is not far behind LoRA. For our translation task with the LLaMA2 7B base model, TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) out performs LoRA ($\mathbf{vBuA}$) while using 12.5% of the number of parameters. This consistent performance illustrates it effectiveness across diverse model scales and task types. The TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) configuration, while marginally outperformed by TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) , is notable for its parameter efficiency. This method achieves comparable performance to TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) for typical ranks $r = 8, 16, 32, 64, 128$, but with a reduced parameter count.

Both TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) show robust performance at the same rank where traditional LoRA ($\mathbf{vBuA}$) is optimized (i.e., performed best). This implies that for systems pretuned for LoRA ($\mathbf{vBuA}$) , TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) can be utilized with the same rank configuration as a "drop-in" replacement.

On average, we observe a 1.36 % decline in TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) performance compared to the LoRA ($\mathbf{vBuA}$) model with the LLaMA 2 7B model. This decline, however, is marginally higher at 1.95 % with the 2B model. These findings suggest that the efficiency of TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) configuration may enhance with a larger or more capable base models (such as the LLaMA2 70B model). This hypothesis warrants a future exploration which we leave for future research.

4.1 Task-Dependent Optimal Rank

From Table 2, we can see that the optimal rank for LoRA ($\mathbf{vBuA}$) varies significantly across different tasks. Furthermore, perhaps surprisingly, a higher rank does not result in higher scores. For example, for LoRA ($\mathbf{vBuA}$) , a rank of 2 suffices for achieving best performance for the Squad task, while a higher rank of 64 is optimal for GSM8K. In scenarios where traditional LoRA ($\mathbf{vBuA}$) requires a higher rank, Tied-LoRA, especially TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) , present an effective alternative by delivering comparable performance with substantially fewer parameters. For instance, for GSM8K, TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) needs only 4.3 % of the parameters that LoRA ($\mathbf{vBuA}$) uses, while achieving a comparable performance (EM score of 31.77 vs. 32.75 for TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and LoRA, respectively).

4.2 Stability Across Ranks

As indicated by Figures 2a and 2b, apart from TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) , all other Tied-LoRAmethods experience a decline in performance with increase rank. This trend highlights a general challenge faced by Tied-LoRAconfigurations, with TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) being an exception. Specifically, TL$_3$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and TL$_4$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) exhibit the most dramatic drop at higher ranks among all Tied-LoRA configurations. We leave addressing these limitations for future research.

Task	LoRA Layer 1	LoRA Layer 32	TL$_5$ ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$)
DialogSum	37.856	31.688	39.731
GSM8K	13.798	2.729	27.066
HellaSwag	77.644	47.301	91.755
IWSLT2017	38.048	22.465	41.37
Squad	85.326	56.216	87.739

Table 3: LoRA applied to a single layer in the transformer vs. TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) . All results in this table are for a rank of 16 for the 7B base model and use the same number of trainable parameters.

Figures 2c and 2d show only the best Tied-LoRAconfigurations, along with baselines LoRA ($\mathbf{vBuA}$) and Vera ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) . While TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) aligns closely with TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) at typical ranks of $4 - 16$, it also exhibits a small performance reduction at higher ranks. This pattern is repeated for Vera ($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) as well. In contrast, TL$_6$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) maintains high performance across a broad range of ranks and is closest to LoRA ($\mathbf{vBuA}$) .

4.3 Layer Selection Vs. Tied-LoRA

The success of Tied-LoRA, specifically TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) as seen from Table 2, begs the question – Would adding LoRA to a single transformer layer lead to similar performance as TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) ? After all single-layer LoRA and TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) would use the same number of parameters. After all, TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) does not use layer-specific parameters (recall the $\mathbf{v}$ and $\mathbf{u}$ are frozen and set to $\mathbf{1}$) and has the same parameter count as applying LoRA ($\mathbf{vBuA}$) to a single layer in the transformer model. To examine this, we trained all tasks with LoRA applied to a single transformer layer's attention projection matrices. The obvious follow-up question is, which layer should LoRA be applied to? We attempt single-layer LoRA on the lowest (closest to the input embeddings) layer, which we designate as "Layer 1" and the highest layer, "Layer 32" in the LLaMA2 7B model.

Table 3 compares the performances of single-layer LoRA against TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) (which uses the same number of trainable parameters as single-layer LoRA). The TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) configuration is a strong performer on all the tasks we examined,

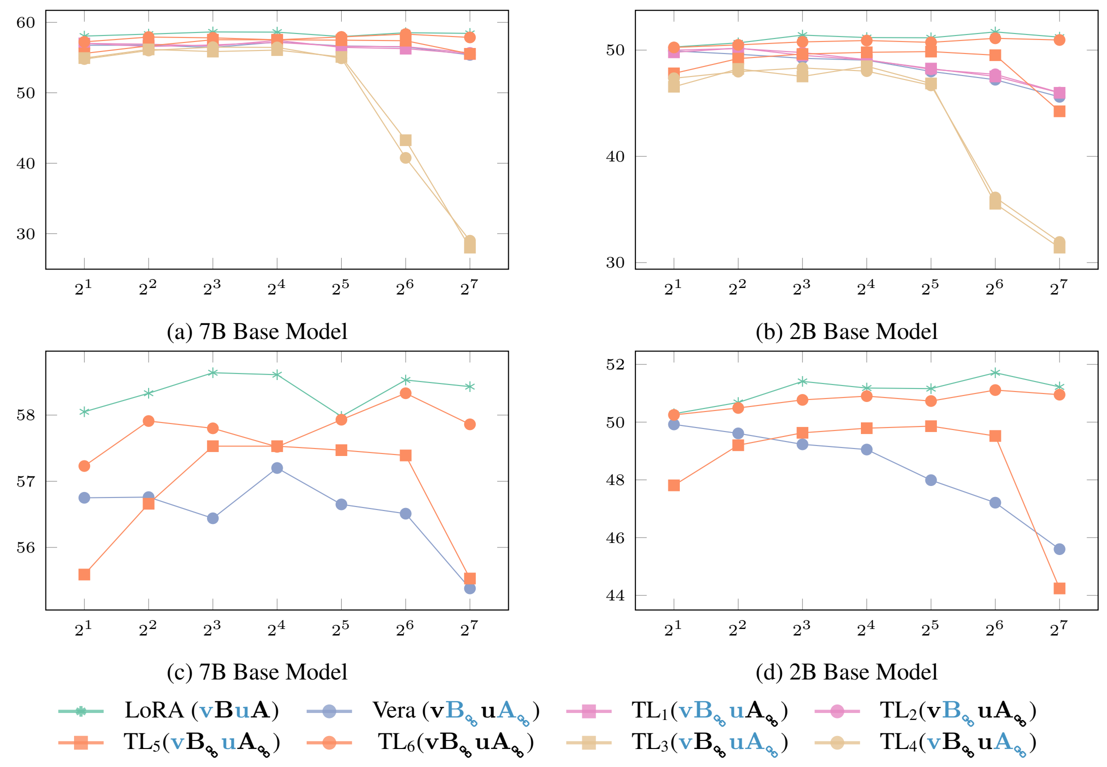


Figure 2: Plots showing the performance of the Tied-LoRAconfigurations averaged over tasks across all ranks.Figures 2a and 2b display all Tied-LoRAconfigurations, while Figures 2c and 2d display the best Tied-LoRAconfigurations with LoRA and Vera as baselines. Appendix A contains plots for each task and base model.

while single-layer LoRA does not perform well. While applying LoRA to Layer 1 is considerably better than applying LoRA to Layer 32 (the highest layer) it still lags behind TL$_5$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) configuration. This suggests that there is potentially a single low-rank update that can be applied to all layers to boost performance, but it is hard to find a low-rank update for a single-layer that results in strong performance.

5 Related Work

Parameter-efficient fine-tuning (PEFT): Recent work on PEFT of pretrained language models has shown competitive capabilities, often matching full fine-tuning performance for task-specific model customization while utilizing significantly fewer trainable parameters. Adapters (Houlsby et al., 2019; Pfeiffer et al., 2021) introduce task-specific parameters within the transformer layers that adapt to a particular task. Prompt tuning based methods such as P-Tuning and Prefix-Tuning (Li and Liang, 2021; Liu et al., 2023) attempt to do the same but via task-specific vectors that can be appended to the inputs or at various layer representations. BitFit and IA3 (Ben Zaken et al., 2022; Liu et al., 2022) are PEFT methods that attempt to only alter bias vectors or scaling vectors in the base large language model.

Low-Rank adaptation (LoRA): One of the most popular PEFT techniques is LoRA, introduced by Hu et al. (2021). LoRA employs low-rank matrix approximations of full weights' gradient-descent (GD) update to significantly reduce the number of trainable parameters. Importantly, LoRA can incorporate the low-rank updates into the frozen base weights after the fine-tuning process, avoiding any inference speed penalties or model architecture changes. In summary, LoRA paves the way for efficient fine-tuning for task-specific customization of large models with minimal computational overhead and no changes to the model's architecture.

Extensions to LoRA: Since its arrival, there have been several efforts to improve the LoRA method. These methods mostly concentrated around reducing the trainable parameters and memory footprint while increasing the performance of the method on downstream tasks. AdaLoRA (Zhang et al., 2023) introduces dynamic rank adjustment for the low-rank matrices during the fine-tuning process. The fundamental premise of this extension is to optimally distribute the parameter budget over model layers. Chavan et al. (2023) combined the adapter tuning with LoRA to derive a generalized framework that utilized both methods for increased flexibility and capability across a wide variety of tasks and datasets. Kopiczko et al. (2023) proposes the VeRA method the freezes randomly initialized projection matrices and introduces trainable scaling vectors that vary across layers. This method shows similar performance to the LoRA ($\mathbf{vBuA}$) method while dramatically reducing the number of trainable parameters. We view VeRA as one specific configuration which lies on end of the Tied-LoRAspectrum.

Tangential to the efforts that aim to reduce trainable parameters, QLoRA (Dettmers et al., 2023), significantly reduces the memory usage of LoRA using a 4-bit or 8-bit quantized base language model during training. The method provides algorithms and custom kernels to backpropagate gradients through the frozen, quantized base model to update low-rank matrices during training, resulting in considerable reduction in memory usage. Combining quantization and reduction in the number of trainable parameters is a direction of future work.

Weight tying: Weight tying (Press and Wolf, 2017; Inan et al., 2017) is a common approach that reduces the number of parameters by using the same set of weights in different parts of the network. Typically the input word embedding layer and the output word embedding layer (sometimes referred to as the language model head) are tied. In this study, we apply weight tying to the low-rank weight matrices used in LoRA, and share them across the layers of the base language model. This simple procedure leads to efficient training methods where the number of trainable parameters are either unaffected by, or only increases marginally with the number of hidden layers. As models get deeper this approach naturally provides greater parameter reduction over original LoRA method.

Vision Transformers: Ideas similar to Tied-Lora are also being explored in the vision based tasks. Dong et al. (2023), for example, uses weight tying and bottleneck adapters.

6 Conclusion & Future Work

In this paper, we introduced the Tied-LoRA paradigm, a novel approach to enhance the parameter efficiency of Lora by employing a simple technique of weight-tying and selective training of low-rank matrices.

Our empirical analysis demonstrates that the $\mathrm{TL}_6(\mathbf{vB}_{\text{🔗}}\mathbf{uA}_{\text{🔗}})$ configuration achieves performance comparable to Lora across various tasks, while utilizing only a fraction of the parameters employed by Lora across a spectrum of low-rank dimensions. This efficiency becomes more pronounced at higher ranks, leading to a more aggressive reduction in the number of trainable parameters compared to Lora. Remarkably, in the translation task, $\mathrm{TL}_6(\mathbf{vB}_{\text{🔗}}\mathbf{uA}_{\text{🔗}})$ surpassed Lora's performance while using only 12.5% of the number of parameters.

Our study highlights that the benefits of this configuration are particularly evident in tasks that leverage the inherent strengths of the base language model, such as commonsense NLI, extractive QA, and summarization. Tasks involving mathematical reasoning and arithmetic calculations, however, favor the sheer learning capacity of Lora with more parameters.

As language models continue to advance, the Tied-LoRA configurations, with their optimized efficiency, emerge as a promising candidate to replace traditional Lora in a broader range of applications. This progression underscores the relevance of Tied-LoRA as a scalable solution in the dynamic landscape of large language model customization.

For future research, we plan to delve into the application of Tied-LoRA methods on larger base models. This exploration aims to assess their scalability and effectiveness within the broader context of large language models. Additionally, we intend to investigate weight tying in other parameter-efficient fine-tuning methods such as Adapters and Prefix Tuning, both of which introduce layer-specific parameters.

Limitations

Parameter Efficient Fine-Tuning methods are inherently sensitive to the base large language which they are applied to as well as the specific customization task. While we attempt to test our methods on multiple base models computation cost restricts us to only 2 models (so far). While extending our analysis to other models is possible, extending to more tasks is more challenging as the variety of tasks is large. Furthermore, predicting the behavior of a PEFT method on a new task based on it's performance on some existing task is very challenging.

Even in our analysis we did not expect translation task to be an outlier (our Tied-LoRAmethod out performed LoRA) because on all the other tasks LoRA was slightly better. Thus, we caution against very strong claims of task generalization and highlight that while we show results on diverse tasks there are still a wide range of tasks we have not explored.

References

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. 2023. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Wei Dong, Dawei Yan, Zhijun Lin, and Peng Wang. 2023. Efficient adaptation of large vision transformer via adapter re-composing. *arXiv preprint arXiv:2310.06234*.

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*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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

Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. 2023. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. *arXiv preprint arXiv:2310.20624*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Xianghui Sun, Yunjie Ji, Baochang Ma, and Xiangang Li. 2023. A comparative study between full-parameter and lora-based fine-tuning on chinese instruction data for instruction following large language model. *arXiv preprint arXiv:2304.08109*.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

A Breakdown of Performances with Ranks

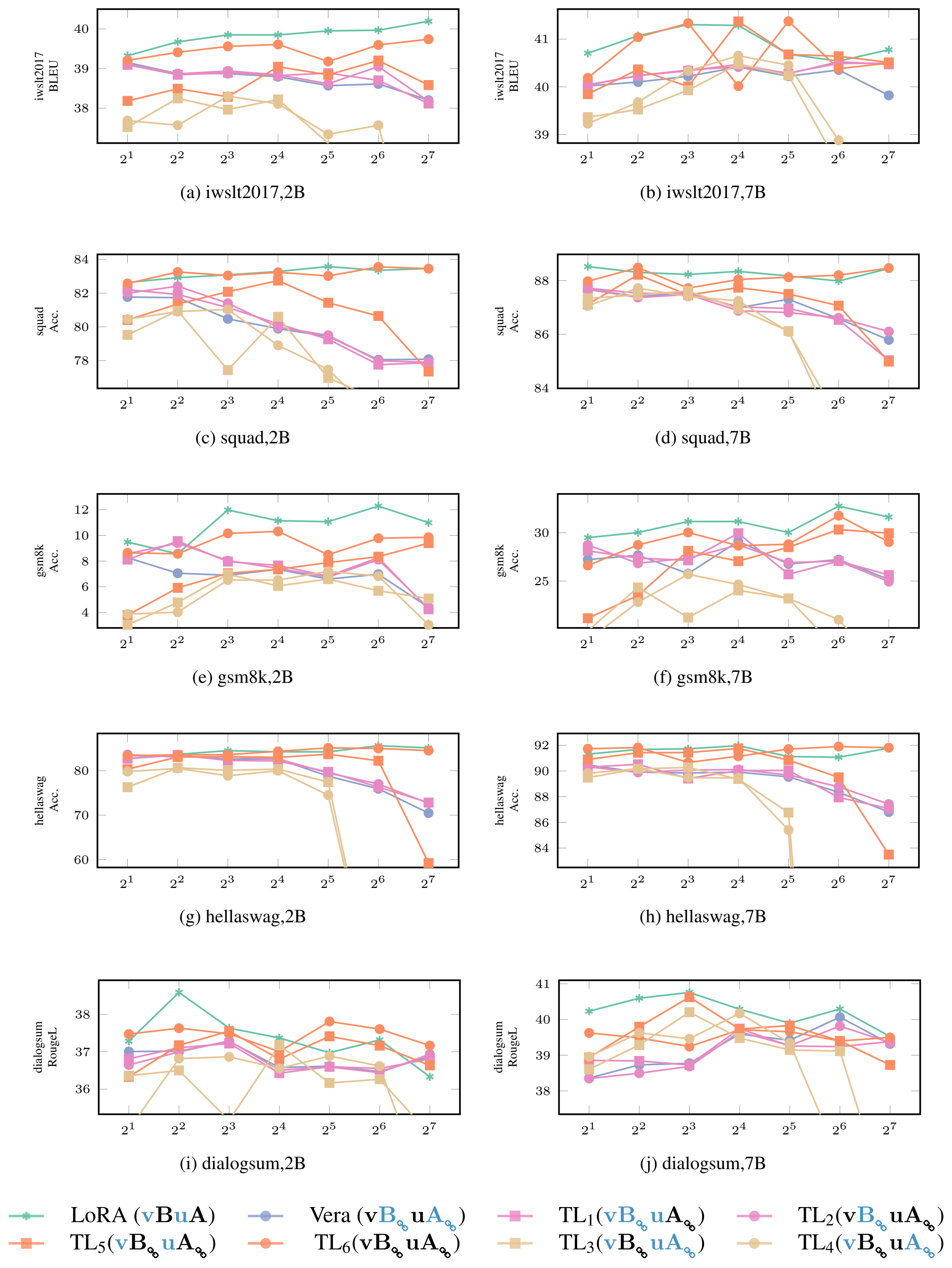


Figure 3: Plots showing the performance of the Tied-LoRAconfigurations along with the baseline LoRA ($\mathbf{vBuA}$) for 5 diverse tasks at 4 different values for low-rank dimension setting. Note that we let the plot for TL$_3$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) and TL$_4$($\mathbf{vB}_{\infty}\mathbf{uA}_{\infty}$) go out of bounds to show details for the other curves.