

000 001 UNoLoRA: SINGLE LOW-RANK ADAPTATION FOR EF- 002 FICIENT MULTITASK FINE-TUNING 003 004

005 **Anonymous authors**

006 Paper under double-blind review

007 008 ABSTRACT 009

011 Recent advances in Parameter-Efficient Fine-Tuning (PEFT) have shown Low-
012 Rank Adaptation (LoRA) to be an effective implicit regularizer for large language
013 models. Building on these findings, we propose UNoLoRA, a novel approach
014 that leverages a single shared LoRA module for efficient multi-task learning.
015 While existing methods typically use separate LoRA adaptations for each task, our
016 approach demonstrates that a single shared adapter can effectively capture both
017 task-specific and task-agnostic knowledge. We further introduce UNoLoRA*, an
018 enhanced variant that employs a shared hypernetwork to generate task-specific
019 embeddings, improving convergence and task adaptation. Our method significantly
020 reduces trainable parameters to just 0.05% per task while maintaining competitive
021 performance on the GLUE benchmark. Our analysis reveals that the A and B
022 matrices in our shared LoRA adapter naturally develop complementary roles: A
023 matrices capture generalizable features across tasks, while B matrices specialize in
024 task-specific representations. Our results show that sharing a single LoRA adapter
025 can achieve efficient multi-task learning while significantly reducing memory
026 requirements, making it particularly valuable for resource-constrained applications.

027 1 INTRODUCTION 028

029 The recent progress of Large Language Models (LLMs) has advanced the field of Natural Language
030 Processing significantly, but their increasing sizes make deployment and adaptation for specific or
031 multiple tasks complicated, making parameter-efficient methods essential. Multi-task learning is
032 advantageous in several ways, such as helping models develop robust and transferable representations,
033 lowering memory usage, and making it easier to adapt to multiple new tasks. However, it comes with
034 a set of challenges, including negative transfer, where learning one task can hurt the performance of
035 the model on other tasks, and the need for more model parameters, which can reduce efficiency.

036 Parameter-Efficient Fine-Tuning (PEFT) methods, particularly Low-Rank Adaptation (LoRA) (Hu
037 et al., 2021), which our work builds upon, have gained attention due to their ability to adapt models
038 to new tasks with minimal overhead. Recent studies have shown that LoRA behaves as an implicit
039 regularizer (Biderman et al., 2024), helping mitigate catastrophic forgetting and maintaining diverse
040 generations, suggesting its suitability for multi-task learning.

041 Building on these insights, we introduce UNoLoRA, a novel approach that uses a single LoRA module
042 for efficient multi-task learning in LLMs. Unlike previous methods that use separate LoRA adapters
043 for each task, UNoLoRA employs a shared LoRA module across all the tasks, capitalizing on LoRA's
044 suspected implicit regularization properties to facilitate knowledge sharing effectively between tasks.

045 This addresses some of the challenges in multi-task learning - it ensures parameter efficiency by using
046 a single low-rank adaptation, minimizing the additional parameters required for multiple tasks. The
047 shared nature of the LoRA module allows for task-agnostic adaptations that mitigate negative transfer
048 between tasks.

049 Furthermore, we introduce UNoLoRA*, which uses a shared hypernetwork to generate task-specific
050 embeddings on top of UNoLoRA, which allows the model to distinguish between different tasks and
051 learn task-specific adaptations.

053 We evaluate UNoLoRA and UNoLoRA* on the GLUE benchmark (Wang, 2018), demonstrating their
effectiveness in a multi-task setting. Our experiments show that UNoLoRA* achieves competitive

054 performance with existing multi-task approaches on GLUE while offering improved parameter
 055 efficiency and having a significantly higher per-step convergence than UnoLoRA.
 056

057 The main contributions of our work are:

- 058 1. A novel single-LoRA-based architecture designed for multi-task learning in LLMs.
 059 2. Comprehensive empirical evaluation of UnoLoRA and UnoLoRA* on GLUE.
 060 3. Analysis and visualization of the behavior and properties of LoRA matrices in single-task
 061 versus multi-task settings.

063 2 RELATED WORK

066 2.1 MULTI-TASK LEARNING IN LARGE LANGUAGE MODELS

068 Multi-task learning (MTL) in Large Language Models (LLMs) has gained attention due to the
 069 potential for improving model generalization and efficiency. Older approaches often involve full
 070 fine-tuning the model on multiple tasks simultaneously (Liu et al., 2019; Aghajanyan et al., 2021).
 071 This can lead to challenges such as negative transfer and increased computational requirements (Wang
 072 et al., 2019).

073 Recent work has explored more parameter-efficient approaches. Adapter based methods (Houlsby
 074 et al., 2019; Pfeiffer et al., 2020) introduce small task specific modules while keeping the base
 075 model frozen. Prompt-tuning techniques (Lester et al., 2021; Li & Liang, 2021) modify the input
 076 representation to adapt models to new tasks.

077 A significant advancement in this area is the HyperFormer approach, introduced by Mahabadi et al.
 078 (2021). This method employs shared hypernetworks for parameter-efficient multi-task fine-tuning of
 079 Transformers. HyperFormer learns to generate task specific adapter parameters, enabling efficient
 080 sharing of knowledge across tasks while maintaining task specific adaptations. This approach
 081 significantly reduces the number of per task trainable parameters compared to traditional adapter
 082 methods, while achieving superior performance on GLUE.

083 2.2 IMPLICIT REGULARIZATION IN NEURAL NETWORKS

085 Regularization in machine learning is essential in preventing overfitting and improving model
 086 generalization. Implicit regularization, which refers to the natural biases of optimization methods
 087 or architectural constraints towards simpler, more generalizable solutions (Neyshabur et al., 2017),
 088 has gained attention in deep learning. Implicit regularization happens without explicit regularization
 089 terms, as seen in several training dynamics and optimization algorithms (Gunasekar et al., 2017).

090 Recent work shows that Low-Rank Adaptation (LoRA) possesses strong implicit regularization
 091 properties. Biderman et al. (2024) found that LoRA’s low-rank structure leads to learning less and
 092 forgetting less compared to full fine-tuning, suggesting it constrains models in ways that mitigate
 093 catastrophic forgetting and may promote positive transfer between tasks.

094 LoRA’s regularization aligns with broader trends in deep learning, where limiting the parameter space
 095 tends to improve generalization. Techniques like pruning (Han et al., 2015) and quantization (Jacob
 096 et al., 2018) achieve model complexity reduction while maintaining performance through implicit
 097 regularization.

099 LoRA’s parameter efficiency proves to be effective in the multi-task learning scenario. By sharing a
 100 single LoRA module across multiple tasks, its regularization properties enable effective fine-tuning,
 101 balancing task-specific adaptations with general language understanding. This shared adaptation
 102 of LoRA both prevents overfitting and promotes generalization. This allows UnoLoRA* to deliver
 103 competitive performance with minimal additional parameters.

104 3 METHODOLOGY

106 **Problem Formulation.** Our focus is on a multi-task learning problem where we seek to develop a
 107 single model capable of performing well across diverse tasks. Consider a pre-trained language model

108 M_θ with parameter set θ , and a collection of target tasks $\mathcal{T} = \{T_1, T_2, \dots, T_K\}$, where K represents
 109 the total number of tasks. Our goal is to determine an optimized set of parameters θ^* that maximizes
 110 performance across all the tasks.

111 For each task $T_j \in \mathcal{T}$, we have a corresponding dataset $D_j = \{(X_j^i, Y_j^i)\}_{i=1}^{M_j}$. Here, X_j^i denotes the
 112 input text, Y_j^i represents the associated label for the i -th instance of the j -th task, and M_j indicates
 113 the total number of samples in task j .

115 3.1 MODEL ARCHITECTURE

116 Our model architecture consists of the pre-trained language model M_θ as the shared backbone
 117 network, enhanced with an UnoLoRA* adaptation module. This module is integrated into the self-
 118 attention and encoder-decoder attention sub-layers of both the encoder and decoder blocks in the
 119 transformer architecture, enabling task-specific adaptations without modifying the original pre-trained
 120 weights.

121 Given a pre-trained weight matrix $W \in \mathbb{R}^{d \times k}$, the UnoLoRA*-adapted weight matrix W' is computed
 122 as:

$$123 W' = W + \alpha \cdot BA \quad (1)$$

124 where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are learnable low-rank matrices shared across all tasks, $r \ll$
 125 $\min(d, k)$ is the rank of the adaptation, with d being the input dimension and k being the output
 126 dimension, and α is a scaling factor (Figure 1).

127 Our PCA visualization comparing the distribution of LoRA matrices A and B in multi-task learning,
 128 as demonstrated in Figure 3, reveals that the A matrix exhibits strong generalization capabilities
 129 across different tasks, while the B matrix captures task-specific features. This finding motivated our
 130 design choice to multiply task-specific information with the A matrix, leveraging its generalization
 131 power to better adapt to new tasks while maintaining task-specific knowledge.

132 To enable the model to distinguish between different tasks and learn task-specific adaptations within
 133 the shared LoRA space, we introduce a Shared Hypernetwork module. This module generates
 134 task-specific embeddings by combining task IDs, sample encodings, and position information:

$$135 e_t = H(t, s, p) \quad (2)$$

136 where H is the Shared Hypernetwork, t is the task ID, s is the sample encoding, and p is the position
 137 information. The output $e_t \in \mathbb{R}^{d_e}$ is a task-specific embedding, where d_e is the dimensionality of the
 138 task embedding space (Figure 2). This unified embedding space is essential for enabling effective
 139 knowledge sharing across tasks while maintaining task-specific characteristics.

140 The inclusion of sample-level encodings is crucial as it allows the model to capture fine-grained,
 141 instance-specific features that may be relevant across multiple tasks. Layer-wise position embeddings
 142 provide important contextual information about how different layers in the transformer architecture
 143 process and transform the input, enabling more nuanced adaptations at different levels of abstraction.
 144 This multi-level representation ensures that the model can adapt its behavior based on both the specific
 145 requirements of each input sample and its position in the network hierarchy.

146 The Shared Hypernetwork consists of several components: A bottleneck network that processes the
 147 sample encodings:

$$148 b = B(s) \quad (3)$$

149 where B is a multi-layer perceptron and $b \in \mathbb{R}^{d_b}$ is the bottleneck representation. This bottleneck
 150 architecture is crucial for distilling high-dimensional sample encodings into a compact, information-
 151 rich representation that captures essential features while reducing computational overhead.

152 Task and position embeddings:

$$153 e_{task} = E_{task}(t), \quad e_{pos} = E_{pos}(p) \quad (4)$$

154 where E_{task} and E_{pos} are embedding layers. These dedicated embedding spaces allow the model to
 155 learn distinct representations for task identity and structural position, ensuring that both task-specific
 156 requirements and architectural context are properly encoded.

162 A network that processes the concatenated position and task embeddings (Xiao et al., 2023) to
 163 differentiate between transformer blocks, and between the query and the value LoRA adapters:
 164

$$e_t = C([e_{task}, e_{pos}, b]) \quad (5)$$

166 where C is a multi-layer perceptron and $[\cdot, \cdot, \cdot]$ denotes concatenation. This fusion network is essential
 167 for learning complex interactions between task, position, and sample-specific information, creating a
 168 unified representation that captures all relevant aspects of the current adaptation context.
 169

170 The task-specific embedding e_t is then used to generate scaling factors that modulate the A matrix in
 171 the UnoLoRA* module:

$$s_t = S(e_t) \quad (6)$$

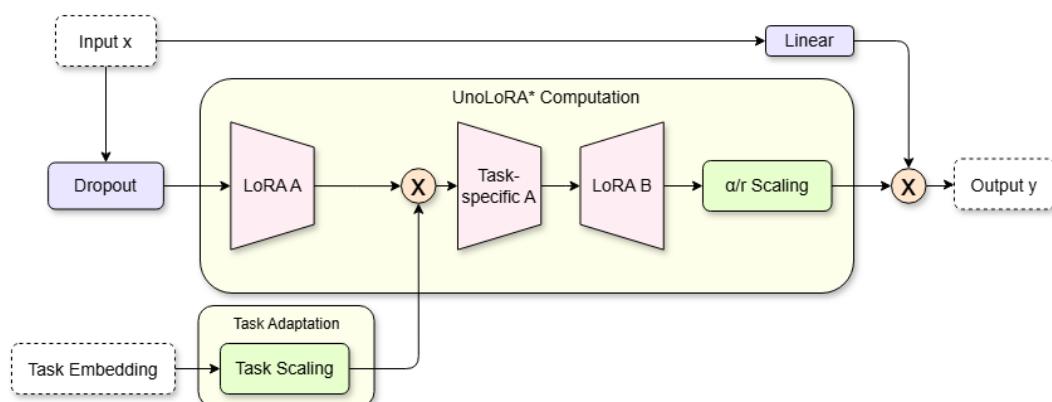
172 where $S : \mathbb{R}^{d_e} \rightarrow \mathbb{R}^r$ is a linear layer that projects the task embedding to the LoRA rank dimension.
 173 This projection layer plays a critical role in translating the rich task-specific information into
 174 appropriate scaling factors that can directly influence the LoRA adaptation process.
 175

176 The task-specific scaling factors are applied to the LoRA adaptation:

$$W' = W + \alpha \cdot B(A \cdot \text{diag}(s_t)) \quad (7)$$

177 where $\text{diag}(s_t)$ creates a diagonal matrix from the scaling vector s_t . By applying the scaling to the
 178 A matrix, we leverage its demonstrated generalization capabilities (Figure 3) while maintaining the
 179 task-specific adaptations learned by the B matrix.
 180

181 This approach provides a sophisticated mechanism for multi-task adaptation, combining the generalization
 182 power of the A matrix with fine-grained task-specific information from the Shared
 183 Hypernetwork. The integration of sample-level encodings and position information enables the model
 184 to capture both instance-specific features and layer-wise contextual information, resulting in more
 185 effective and nuanced adaptations across different tasks.
 186



202
 203 Figure 1: UnoLoRA* Computation: Illustration of how UnoLoRA* modifies the weight matrix W
 204 using low-rank adaptation matrices A and B . The scaling factor α and task-specific scaling vector s_t
 205 allow for task-dependent adjustments.

208 3.2 TRAINING OBJECTIVE

209 We optimize the model parameters θ , the shared LoRA matrices B and A , the task embeddings $\{e_i\}$,
 210 and the hypernetwork using AdamW (Loshchilov & Hutter). The optimization process jointly learns
 211 the shared LoRA adaptation, task-specific scaling factors to maximize the overall performance across
 212 all tasks.
 213

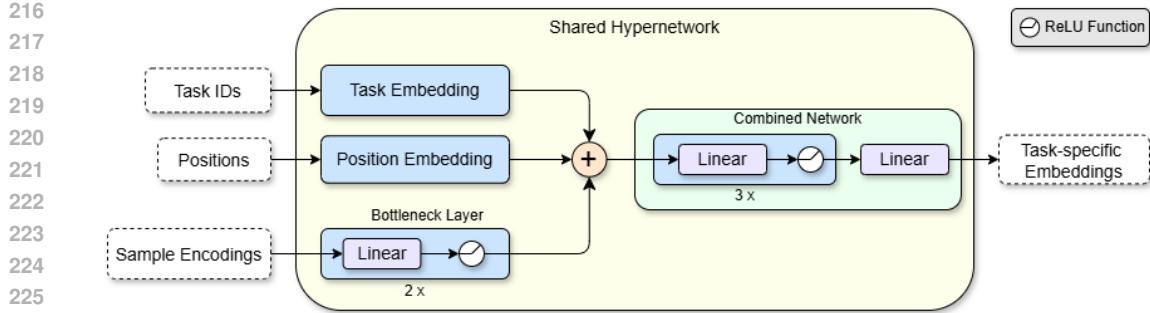


Figure 2: Shared Hypernetwork: The Shared Hypernetwork generates embeddings that encode task-specific information. These embeddings are used to adapt the LoRA weights dynamically, ensuring that task-specific nuances are captured effectively.

3.3 INTEGRATION WITH PRE-TRAINED MODELS

The integration of our UnoLoRA adapters with pre-trained language models is achieved through a wrapper architecture, which we call `EnhancedUnoLoRAWrapper`. This wrapper encapsulates a pre-trained T5 model and augments it with task-specific adaptations while preserving the original model’s parameters. Given a pre-trained T5 model M_θ , we replace specific linear layers in the self-attention and cross-attention modules with our UnoLoRA layers. The replacement occurs in both the encoder and decoder blocks:

$$M_\theta = \text{Replace}(M_\theta, \text{UnoLoRA}) \quad (8)$$

where Replace is a function that substitutes the query (Q) and value (V) projections in each attention layer with UnoLoRA modules. The UnoLoRA module extends the standard LoRA adaptation by incorporating task-specific scaling:

$$W't = W + \alpha \cdot (B \cdot \text{diag}(s_t))A \quad (9)$$

where $W't$ is the task-specific adapted weight matrix, W is the original weight matrix, B and A are the LoRA matrices, α is the scaling factor, and s_t is the task-specific scaling vector generated by the Shared Hypernetwork. The integration process involves the following steps:

1. Freezing the base model parameters:

$$\forall \theta \in M_\theta : \frac{\partial \mathcal{L}}{\partial \theta} = 0 \quad (10)$$

2. Replacing attention layers with UnoLoRA modules:

$$\text{AttnUnoLoRA} = \text{UnoLoRA}(\text{Attnoriginal}, r, \alpha, d_e) \quad (11)$$

where r is the LoRA rank, α is the scaling factor, and d_e is the task embedding dimension.

3. Initializing the Shared Hypernetwork:

$$H = \text{Shared Hypernetwork}(|\mathcal{T}|, d_h, d_e, d_s, d_b, L, p_{\max}) \quad (12)$$

where $|\mathcal{T}|$ is the number of tasks, d_h is the hidden dimension, d_e is the output dimension, d_s is the sample encoding dimension, d_b is the bottleneck dimension, L is the number of layers, and p_{\max} is the maximum position.

This integration approach allows for efficient task-specific adaptation of the pre-trained model while maintaining its original knowledge. The UnoLoRA modules and Shared Hypernetwork introduce a relatively small number of trainable parameters, enabling rapid adaptation to new tasks without the need for full model fine-tuning.

270 **4 EXPERIMENTS**
 271

272 **Model and Implementation:** We use T5-base (Raffel et al., 2020) as our backbone model across
 273 all experiments. For multi-task experiments, we implement UnoLoRA, a single custom version of
 274 LoRA (Hu et al., 2021) shared across all tasks. Furthermore, we implement UnoLoRA* which is
 275 UnoLoRA with a shared hypernetwork to generate task specific embeddings to aid the LoRA learning
 276 process. For single-task experiments, we utilize the standard LoRA implementation from Hugging
 277 Face’s Transformers (Wolf et al., 2020).

278 **Datasets and Evaluation:** We evaluate our models on the GLUE (General Language Understanding
 279 Evaluation) benchmark (Wang, 2018), which includes tasks such as Natural Language Inference
 280 (MNLI), Sentiment Analysis (SST-2), Paraphrase Detection (MRPC, QQP), Textual Similarity (STS-
 281 B), Grammar Acceptance (CoLA), and Question Answering (QNLI, RTE), following the approach
 282 of Raffel et al. (2020). Since the original test sets are not publicly available, we adopt the data split
 283 strategy from Zhang et al. (2020). For smaller datasets (RTE, MRPC, STS-B, CoLA) with fewer
 284 than 10K samples, we split the original validation set equally into validation and test sets. For larger
 285 datasets, we create a validation set by reserving 1K samples from the training data and use the original
 286 validation set for testing.

287 **Baselines:** We compare our method against several baselines, with careful consideration of how
 288 performance is measured and aggregated:

- 291 • **Single-Task Fine-Tuning:** Independent fine-tuning of T5-base for each task, updating all
 292 parameters.
- 293 • **Single-Task LoRA:** Independent LoRA adaptation for each task, resulting in separate task-
 294 specific adapters. The reported performance reflects the evaluation of a single task-specific
 295 adapter.
- 296 • **Multi-Task Fine-Tuning:** Simultaneous fine-tuning of T5-base on all tasks, updating all
 297 parameters.
- 298 • **HyperFormer++:** Implementation of the enhanced HyperFormer++ (Mahabadi et al., 2021)
 299 approach for multi-task learning with T5-base.

301 For single-task models, best checkpoint is taken for each task(one model per-task). For multi-task
 302 models, performance is measured using a single best checkpoint selected based on average validation
 303 performance across tasks.

305 **Experimental Details:** All experiments use the GLUE benchmark’s natural language understanding
 306 tasks. For multi-task training, we employ temperature-based sampling ($T=10$) to balance task
 307 representation. We train for 50 epochs on smaller datasets and 10 epochs on larger datasets during
 308 single-task LoRA fine-tuning. Following Raffel et al. (2020), we use a constant learning rate of
 309 $1e - 4$ and train for $2^{18} = 262144$ steps, saving checkpoints every 29535 steps. Unlike Raffel et al.
 310 (2020), who report results using task-specific best checkpoints, we adopt a more realistic approach by
 311 selecting a single checkpoint based on the highest average validation performance across all tasks.
 312 This ensures fair comparison between single-task and multi-task approaches. Detailed hyperparameter
 313 settings are provided in Table 2 (See Appendix A.3). All experiments were conducted using NVIDIA
 314 A100 and H100 GPUs (40GB VRAM).

315 **4.1 RESULTS ON THE GLUE BENCHMARK**
 316

317 We evaluate our proposed UnoLoRA method and its enhanced variant UnoLoRA* against several
 318 baselines on the GLUE benchmark, with results presented in Table 1. Our analysis focuses on both
 319 performance and parameter efficiency across single-task and multi-task training paradigms. The
 320 experiments highlight the effectiveness of our approach in terms of parameter efficiency and its ability
 321 to leverage shared information across tasks through the shared LoRA adapter.

324
 325 Table 1: Results on the GLUE benchmark. For MRPC and QQP, we report accuracy/F1. For STS-B,
 326 we report Pearson/Spearman correlation. For other tasks, we report the standard metric.
 327 Bold indicates best results in multi-task training. *Trained* is the per-task trainable parameters of the model.
 328 †: Results reported directly from Mahabadi et al. (2021).

Model	#Params		CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg.
	Total	Trained									
<i>Single-Task Training</i>											
T5 [†] (full fine-tuning)	8.0×	100%	54.85	92.19	88.18 / 91.61	91.46 / 88.61	89.55 / 89.41	86.49	91.60	67.39	84.67
LoRA	1 + (8 × 0.004)	0.4%	56.10	93.81	84.31 / 84.31	90.44 / 89.84	90.19 / 89.79	86.29	93.56	69.78	84.40
<i>Multi-Task Training</i>											
T5 [†] (full fine-tuning)	1.0×	12.50%	54.88	92.54	90.15 / 93.01	91.13 / 88.07	88.84 / 88.53	85.66	92.04	75.36	85.47
HyperFormer++	1.02×	0.290%	63.73	94.03	89.66 / 92.63	90.28 / 87.20	90.00 / 89.66	85.74	93.02	75.36	86.48
UnoLoRA (<i>Ours</i>)	1.004×	0.049%	50.79	94.61	85.78 / 85.78	89.99 / 89.32	88.63 / 88.44	84.71	92.68	77.70	84.40
UnoLoRA* (<i>Ours</i>)	1.004×	0.050%	56.11	93.92	86.76 / 86.76	89.88 / 89.21	88.52 / 88.69	85.24	93.14	76.26	84.95

339
 340 In the single-task setting, traditional LoRA achieves competitive performance (84.40% average)
 341 compared to full fine-tuning (84.67% average) while training only 0.4% of the parameters. This
 342 establishes a strong baseline for parameter-efficient fine-tuning approaches.

343 In the multi-task setting, our enhanced UnoLoRA* achieves an average score of 84.95%, showing
 344 improvement over the base UnoLoRA’s 84.40%. While HyperFormer++ achieves the highest average
 345 performance (86.48%), both our methods offer compelling parameter efficiency, using just a single
 346 shared adapter approach. UnoLoRA* demonstrates particular strengths on several tasks, notably
 347 improving performance on CoLA (56.11% vs 50.79%), MRPC (86.76% vs 85.78%), and MNLI
 348 (85.24% vs 84.71%) compared to base UnoLoRA.

349 A key advantage of our approaches is their parameter efficiency. Both UnoLoRA and UnoLoRA*
 350 require training only about 0.05% of the total parameters, significantly more efficient than full
 351 fine-tuning (12.50% in multi-task setting) and even HyperFormer++ (0.290%). This efficiency is
 352 particularly important in resource-constrained environments or when scaling to larger models.

353 Both variants show strong performance on classification tasks, with UnoLoRA excelling on SST-2
 354 (94.61%) and UnoLoRA* achieving strong results on QNLI (93.14%). Notably, UnoLoRA achieves
 355 the best RTE performance (77.70%) among all multi-task approaches, while UnoLoRA* provides
 356 more consistent performance across the full task suite.

358 5 ANALYSIS

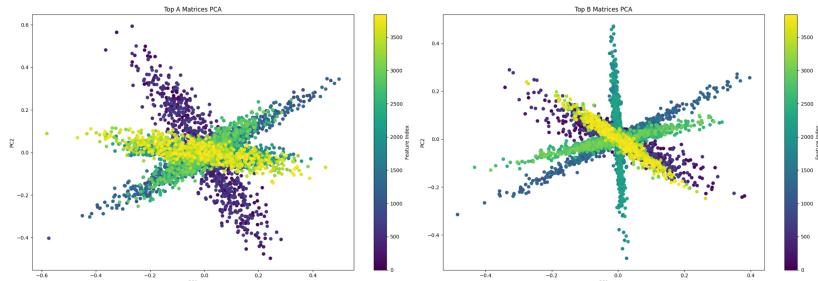
360 We present an analysis of UnoLoRA’s performance and internal mechanisms, demonstrating its
 361 effectiveness in multi-task learning settings through three key findings: (1) superior parameter
 362 efficiency - as discussed earlier, (2) distinct functional specialization in the A and B matrices, and (3)
 363 faster convergence through our enhanced UnoLoRA* variant.

365 5.1 EMPIRICAL ANALYSIS AND FINDINGS

367 To understand how UnoLoRA achieves efficient multi-task learning, we analyzed the properties of
 368 its LoRA matrices through multiple mathematical perspectives and discovered a clear functional
 369 specialization between components:

370 **Matrix Properties and Representations:** Our analysis reveals distinct characteristics between A
 371 and B matrices. The A matrices demonstrate higher singular values and eigenvalues (Figure 5c, 5d),
 372 indicating they capture a broader range of transformations in the parameter space. This is further
 373 supported by their scattered distribution in PCA visualization (Figure 3, left), suggesting diverse
 374 feature representations. In contrast, B matrices show more concentrated eigenvalue distributions and
 375 form dense clusters in PCA space (Figure 3, right), indicating more specialized transformations.

378
379
380
381
382
383
384
385
386
387
388



389 Figure 3: 2D PCA visualization comparing the distribution of LoRA matrices in multi-task learning.
390 A matrices (left) exhibit a dispersed pattern with greater variance suggesting diverse, generalizable
391 features. B matrices (right) show tighter clustering indicating task-specific feature specialization.
392 Points represent individual matrix components projected onto the first two principal components.

393
394

395 **Layer-wise Behavior:** Examining the cross-layer relationships, we observe that A matrices ex-
396 hibit noticeable correlation across different layers (Figure 7, left), suggesting they learn consistent
397 transformations throughout the network. This layer-wise generalization capability, combined with
398 their diverse representational properties, makes them particularly suitable for multi-task learning. B
399 matrices, conversely, show minimal cross-layer correlation (Figure 7, right), aligning with their role
400 in task-specific adaptations.

401 This complementary behavior enables efficient multi-task learning through:

402 **Enhanced Representational Capacity:** Multi-task adaptations demonstrate consistently higher
403 effective rank across all layers (Figure 5a), particularly pronounced in the encoder layers. As shown
404 in Figure 4a, this enhanced representational capacity translates to superior performance scaling,
405 where our multi-task approach (green dots) maintains efficiency across different parameter regimes
406 compared to single-task models (red dots) and either matches or surpasses their performance.

407 **Optimal Parameter Updates:** The multi-task setting exhibits larger Frobenius norms (Figure 5b) and
408 consistently higher singular values (Figure 5c) and eigenvalues (Figure 5d) compared to single-task
409 counterparts. The larger Frobenius norms indicate stronger overall weight updates, suggesting the
410 model makes more substantial adaptations to accommodate multiple tasks. The higher singular and
411 eigenvalues reveal that these adaptations utilize a broader range of transformation directions in the
412 parameter space, allowing the model to capture more complex patterns.

413

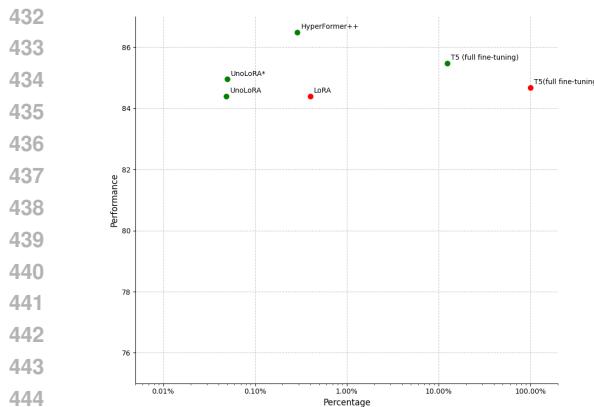
414 5.2 CONVERGENCE ANALYSIS

415

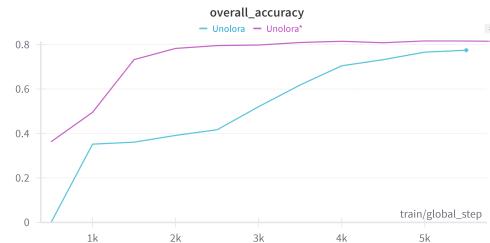
416 Building on these insights, we developed UnoLoRA*, which enhances the base architecture’s ability
417 to learn task-specific features more efficiently. As demonstrated in Figure 4b, UnoLoRA* achieves
418 higher performance at earlier training stages compared to the original UnoLoRA. This pattern of
419 faster convergence is consistently observed across multiple tasks in the GLUE benchmark, including
420 MNLI, STS-B, QQP, and SST-2 (see Appendix A.1 for detailed per-task convergence plots). This
421 faster convergence is particularly valuable in resource-constrained scenarios and rapid deployment
422 settings.

423 The improved early-stage performance can be attributed to the enhanced architecture’s ability to
424 better leverage the functional specialization we observed between A and B matrices, allowing for
425 more efficient learning of both shared and task-specific features.

426
427
428
429
430
431

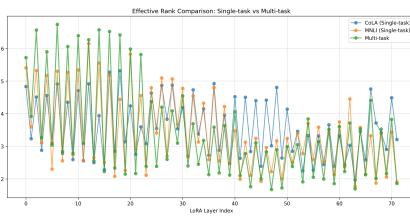


(a) Comparison of accuracy versus percentage (log scale) of trained parameters across different models in a multi-task learning setting. The green dots represent multi-task fine-tuned models, and the red dots represent single-task fine-tuned models.

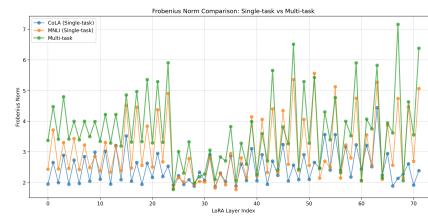


(b) Comparison of the overall accuracy of UnoLoRA and UnoLoRA* over the first 5000 training steps on a subset of the validation dataset. We can see that UnoLoRA* is able to achieve a higher performance at an earlier stage than UnoLoRA.

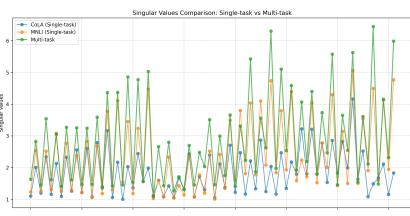
Figure 4: Performance analysis of multi-task learning models. (a) Illustrates the trade-off between model accuracy and the percentage of trained parameters across various models. (b) Shows the per-step convergence rates of UnoLoRA and UnoLoRA*, highlighting the improved early-stage performance of UnoLoRA*.



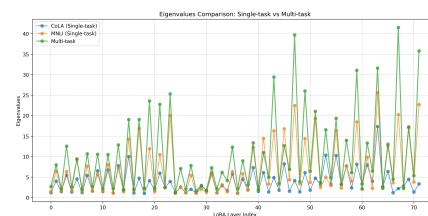
(a) Effective rank comparison across model layers, showing consistently higher values for multi-task training.



(b) Layer-wise Frobenius norm distribution, indicating magnitude of weight adjustments in adaptation matrices.



(c) Distribution of singular values across layers, reflecting the complexity of learned transformations.



(d) Eigenvalue distributions showing the dimensionality of learned feature spaces.

Figure 5: Comparative analysis of LoRA adaptation matrices between single-task (CoLA, MNLI) and multi-task models across different metrics. Layer indices (x-axis) correspond to the model architecture progression from encoder (lower indices) to decoder (higher indices). The multi-task model consistently demonstrates higher effective rank and more distributed eigenvalue patterns, suggesting more complex and comprehensive feature representations compared to single-task variants. This analysis spans multiple mathematical perspectives: effective rank (measuring dimension utilization), Frobenius norm (capturing overall adaptation magnitude), and spectral properties (singular and eigenvalues) revealing the internal structure of learned transformations.

486 6 LIMITATIONS AND FUTURE WORK

488 While UnoLoRA demonstrates promising results in multi-task learning with parameter-efficient fine-
 489 tuning, several limitations and opportunities for future research remain. Firstly, our evaluation mainly
 490 focuses on the GLUE (Wang, 2018) benchmark. While the dataset is comprehensive, evaluating
 491 UnoLoRA on additional datasets would further reinforce the results obtained.

492 Our experiments have been conducted exclusively with the T5-base model (Raffel et al., 2020),
 493 which uses an encoder-decoder architecture. Future work could investigate UnoLoRA’s effectiveness
 494 with other architectural paradigms, such as encoder-only (e.g., BERT) and decoder-only (e.g., GPT)
 495 models. Additionally, testing UnoLoRA across different model scales, both smaller and larger, would
 496 provide valuable insights into its scalability and efficiency characteristics.

497 Several promising directions emerge for future research. First, investigating UnoLoRA’s performance
 498 in few-shot learning scenarios would help understand its effectiveness with limited training data.
 499 Second, exploring task transfer capabilities, particularly between unrelated domains, would provide
 500 insights into cross-domain generalization. Third, extending UnoLoRA beyond natural language
 501 processing to other modalities such as computer vision and audio processing would evaluate its
 502 broader applicability. Furthermore, the shared nature of our single LoRA module presents unique
 503 opportunities for interpretability research. Unlike models with separate task-specific modules, our
 504 approach could enable better analysis of how different tasks influence the learned weights, potentially
 505 providing insights into task relationships and knowledge transfer mechanisms. These extensions
 506 would help establish the boundaries of UnoLoRA’s capabilities and potentially reveal new applications
 507 for parameter-efficient multi-task learning.

508 7 CONCLUSION

511 This paper introduces UnoLoRA, demonstrating that a single shared LoRA module can effectively
 512 handle multi-task learning while requiring only 0.05% trainable parameters per task. Unlike traditional
 513 approaches that require separate LoRA modules for each task, our approach only requires one LoRA
 514 module for all tasks, and achieves competitive performance on the GLUE benchmark.

515 We further enhance this architecture with UnoLoRA*, which employs a shared hypernetwork to
 516 generate task-specific embeddings. This enhancement leads to significantly faster convergence across
 517 multiple GLUE tasks, making it particularly valuable for resource-constrained scenarios and rapid
 518 deployment settings. Our empirical analysis reveals how UnoLoRA achieves efficient multi-task
 519 learning through complementary roles of its components: A matrices capture generalizable features
 520 with consistent cross-layer transformations, while B matrices handle task-specific adaptations.

521 These findings establish UnoLoRA as a promising PEFT method for multi-task learning. The
 522 success of our approach in maintaining performance while drastically reducing parameters opens
 523 new possibilities for efficient model adaptation and has the potential to inspire further research in
 524 PEFT methods, particularly in scenarios where computational resources are limited.

526 REFERENCES

- 528 Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and
 529 Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint*
 530 *arXiv:2101.11038*, 2021.
- 531 Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings,
 532 Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less.
 533 *arXiv preprint arXiv:2405.09673*, 2024.
- 535 Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro.
 536 Implicit regularization in matrix factorization. *Advances in neural information processing systems*,
 537 30, 2017.
- 538 Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for
 539 efficient neural network. *Advances in neural information processing systems*, 28, 2015.

- 540 Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe,
 541 Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for
 542 nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.
- 543 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
 544 and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint*
 545 *arXiv:2106.09685*, 2021.
- 546 Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig
 547 Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient
 548 integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and*
 549 *pattern recognition*, pp. 2704–2713, 2018.
- 550 Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt
 551 tuning. *arXiv preprint arXiv:2104.08691*, 2021.
- 552 Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv*
 553 *preprint arXiv:2101.00190*, 2021.
- 554 Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for
 555 natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.
- 556 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference*
 557 *on Learning Representations*.
- 558 Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-
 559 efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint*
 560 *arXiv:2106.04489*, 2021.
- 561 Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generaliza-
 562 tion in deep learning. *Advances in neural information processing systems*, 30, 2017.
- 563 Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder,
 564 Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv*
 565 *preprint arXiv:2007.07779*, 2020.
- 566 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi
 567 Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text
 568 transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- 569 Alex Wang. Glue: A multi-task benchmark and analysis platform for natural language understanding.
 570 *arXiv preprint arXiv:1804.07461*, 2018.
- 571 Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding
 572 negative transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern*
 573 *recognition*, pp. 11293–11302, 2019.
- 574 Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi,
 575 Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art
 576 natural language processing. In *Proceedings of the 2020 conference on empirical methods in*
 577 *natural language processing: system demonstrations*, pp. 38–45, 2020.
- 578 Zedian Xiao, William Held, Yuchen Liu, and Diyi Yang. Task-agnostic low-rank adapters for unseen
 579 english dialects. *arXiv preprint arXiv:2311.00915*, 2023.
- 580 Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. Revisiting few-sample
 581 bert fine-tuning. *arXiv preprint arXiv:2006.05987*, 2020.
- 582

583 **A APPENDIX**

584 **SUPPLEMENTARY MATERIAL**

585
 586 Please find the link attached to replicate our experiments: Google Drive Link

A.1 TASK CONVERGENCE PLOTS

To investigate how UnoLoRA^{*}'s hypernetwork affects training dynamics, we compare learning curves against the base UnoLoRA model across four representative tasks from the GLUE benchmark. Figure 6 presents the convergence plots for STS-B, SST-2, MNLI, and QQP, showcasing tasks with varying complexity and objectives.

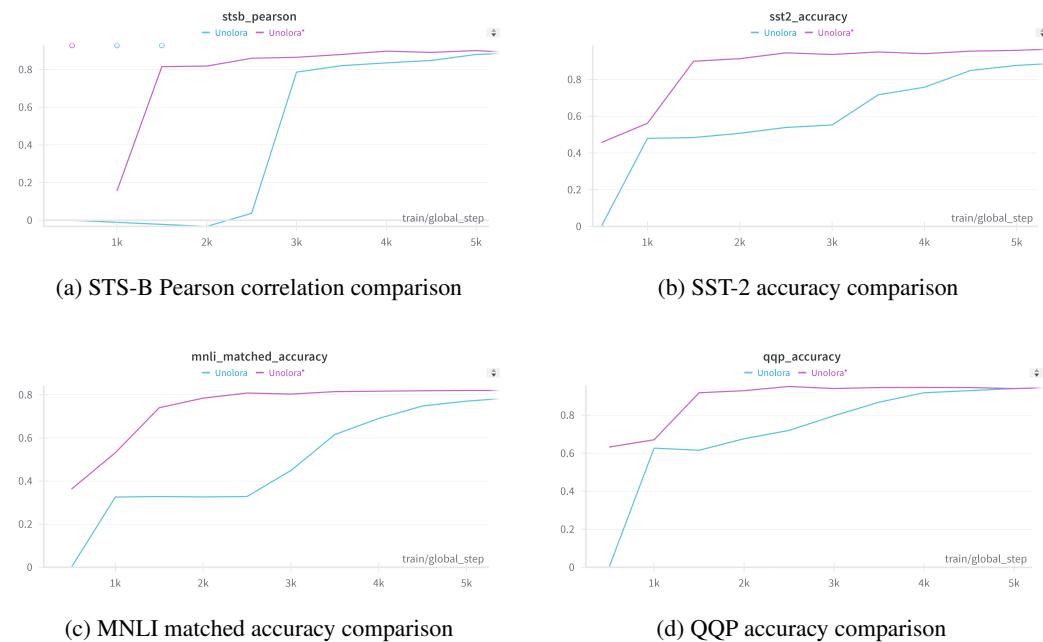
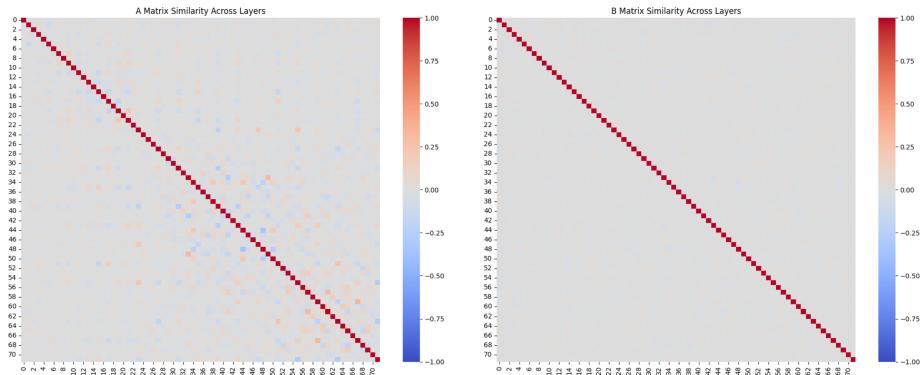


Figure 6: Convergence plots for (a) STS-B Pearson correlation, (b) SST-2 accuracy, (c) MNLI matched accuracy, and (d) QQP accuracy. Each plot compares the performance of UnoLoRA and UnoLoRA* over training steps. UnoLoRA* consistently demonstrates faster convergence and better early-stage performance across all tasks, regardless of the task complexity or evaluation metric.

The plots demonstrate that UnoLoRA*'s improved convergence is not task-specific but rather a general characteristic of the enhanced architecture. This is particularly evident in more complex tasks like MNLI (natural language inference) and QQP (semantic similarity), where the performance gap between UnoLoRA* and the base UnoLoRA is more pronounced in the early stages of training. Even for simpler tasks like SST-2 (sentiment analysis), UnoLoRA* maintains its advantage in convergence speed while achieving comparable final performance.

648
649

A.2 LORA MATRICES CORRELATION ACROSS LAYERS

650
651
652
653
654
655
656
657
658
659
660
661
662663
664
665
666
667
668
669
670
671
Figure 7: Pearson correlation analysis of LoRA matrices across network layers in multi-task learning. Left: A matrices show noticeable correlation between different layers suggesting these matrices learn similar transformations across layers. This consistency across layers indicates the learning of general features that are reused throughout the network, supporting their role in capturing transferable knowledge. Right: B matrices show minimal correlations between layers implying each layer learns distinct transformations, consistent with their role in capturing task-specific adaptations.672
673
674
675

A.3 SAMPLING STRATEGIES FOR MULTI-TASK LEARNING

676
677
678
679
680
681

In multi-task learning, the sampling strategy plays a crucial role in determining the proportion of data from each task that the model is trained on. The goal is to strike a balance between providing enough data for the model to learn each task effectively while avoiding over-training on any particular task. Several sampling strategies have been proposed to address this challenge:

682
683
684
685
686
687
688
689
690
691

Examples-Proportional Mixing: This strategy samples examples from each task in proportion to the size of its dataset. It is equivalent to concatenating all the datasets and randomly sampling examples from the combined dataset. However, when there is a significant disparity in dataset sizes, such as the inclusion of a large unsupervised task, this approach can lead to under-training on the supervised tasks. To mitigate this issue, an artificial "limit" can be set on the dataset sizes before computing the proportions.

692
693
694
695
696
697
698
699
700
701

Temperature-Scaled Mixing: Temperature scaling is another way to address the imbalance in dataset sizes. In this approach, the mixing rates of each task are raised to the power of the reciprocal of a temperature parameter T and then renormalized. When $T = 1$, it is equivalent to examples-proportional mixing. As T increases, the mixing proportions become closer to equal mixing. This allows for adjusting the influence of larger datasets while still considering their relative sizes. The `MultiTaskBatchSampler` used in Mahabadi et al. (2021) falls under this category of temperature-scaled mixing. It aims to balance the proportions of tasks in each batch by sampling tasks according to their dataset sizes. However, this approach can still lead to oversampling of smaller datasets like RTE, as the proportions are solely based on the dataset sizes without considering other factors such as task difficulty or model performance.

During multi-task training, we sample tasks with conventional temperature-based sampling, using a temperature of $T = 10$, following . Tasks are sampled proportionally to $p_\tau^{1/T}$, where $p_\tau = \frac{N_\tau}{\sum_{i=1}^T N_\tau}$ and N_τ is the number of training samples for the τ -th task.

Equal Mixing: In this strategy, examples are sampled from each task with equal probability, regardless of the dataset sizes. While this ensures equal representation of all tasks, it may lead to overfitting on low-resource tasks and underfitting on high-resource tasks.

A.4 HYPERPARAMETERS

Table 2: Hyperparameter settings of T5-base models on GLUE for UnoLoRA and UnoLoRA*

Hyperparameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Rank r				8				
Alpha				16				
Layer L			All Q,V Self-Attention					
Bottleneck dim L				8				
Sample encoding dim L				512				
Dropout				0.1				
Optimizer			AdamW					
Learning Rate				1e-4				
Weight decay				0.01				
Warmup steps				1000				
Max steps			262144					
LR scheduler			Cosine Annealing					
Batch size			32					
Epochs	10	10	50	50	10	10	50	50