Less Data, Less Capacity, More Insights: Rethinking LoRA for Domain-Specific Ultra-Small Data Regimes

Anonymous ACL submission

Abstract

Generating natural language instructions from limited action sequences of an assembly line is critical for human-robot collaboration in manufacturing. However, fine-tuning large language models(LLMs) for this task is computationally prohibitive on small datasets. We present a parameter-efficient approach to adapt a small language model(SLM) for assembly instruction generation. Rather than fine-tuning all parameters, we freeze the base model parameters and finetune selected parameters with help of Low-Rank-Adaptation(LoRA) adapter. This reduces training computations and prevents overfitting on limited dataset. A comprehensive ablation study was conducted across six-dimension and we identify the optimal LoRA configuration: rank (r), scaling Factor (α), dropout, target modules and training epochs using diverse parameter search space. Crucially, we discover that the full LoRA module adaptation achieves the best BLEU score of 30.56%, while the `gate_up_proj` layer alone achieves 97% of this performance with only 25% of the trainable parameters. In our approach use Assembley101 datasets action sequences and achieves 62.88% ROUGE-1, 30.56% BLEU, and 69.53% SBERT similarity. We release code and trained LoRA adapters to enable reproducible research on small datasets.

1 Introduction

The automated translation of fine-grained physical actions into human language is a fundamental challenge at the intersection of computer vision and computational linguistics, intelligent systems move from passive observation to active collaboration (Arvind et al., 2026), the ability to generate natural language and real-time human guidance (Hua et al., 2025) are essential for human-robot interaction (HRI), assistive technologies, and AI enabled industrial automation (Tellex et al., 2020; Yadollahi et al., 2024), In human-robot team, focusing on explainability (Setchi et al., 2020; Dehkordi et al., 2021). The temporally-aligned task actions, often referred as "**Action-to-Text**" mapping, is a cornerstone for creating explainable AI interfaces in assembly environments (Patalas-Maliszewska et al., 2021; Farooqui et al., 2025). Example an industrial assembly task: while a robot or computer vision system, able to recognize a sequence of primitive temporal-segments (Sener et al., 2022; Stengel-Eskin et al., 2022; Arvind et al., 2026) such as *pick_up_screwdriver*, *align_screw*, and *fasten_chassis* (Lea et al., 2017; Farha and Gall, 2019), but a human operator requires a cohesive and fluent instruction: "*Pick up the screwdriver, align the screw with the chassis, and then fasten it collectively.*" Despite significant progress in video captioning and action recognition (Carreira and Zisserman, 2017; Damen et al., 2022; Grauman et al., 2022), generating action to instructions from fine-grained action sequences remains challenging due to three primary limitations in existing research:

(i) Current approaches overlooked domain-specific Ultra-Small-Data Regime; While Low-Rank Adaptation (LoRA) (Hu et al., 2022) enables parameter-efficient fine-tuning, but optimal hyperparameters for ultra-small datasets remains unexplored.

(ii) Fine-tuning of full-parameters is computationally prohibitive and prone to catastrophic forgetting with limited data (Stengel-Eskin et al., 2022; Touvron et al., 2023; Brown et al., 2020). While LoRA reduces trainable parameters, but the most parameter-efficient configuration for ultra-small datasets is unknown.

(iii) Traditional n-gram metrics like BLEU and ROUGE (Papineni et al., 2002; Lin, 2004) fail to account for semantic correctness and functional safety (Meltzer et al., 2024; Post, 2018), which are critical in some domains. For detailed theoretical analysis of LoRA's generalization properties, see Appendix C.

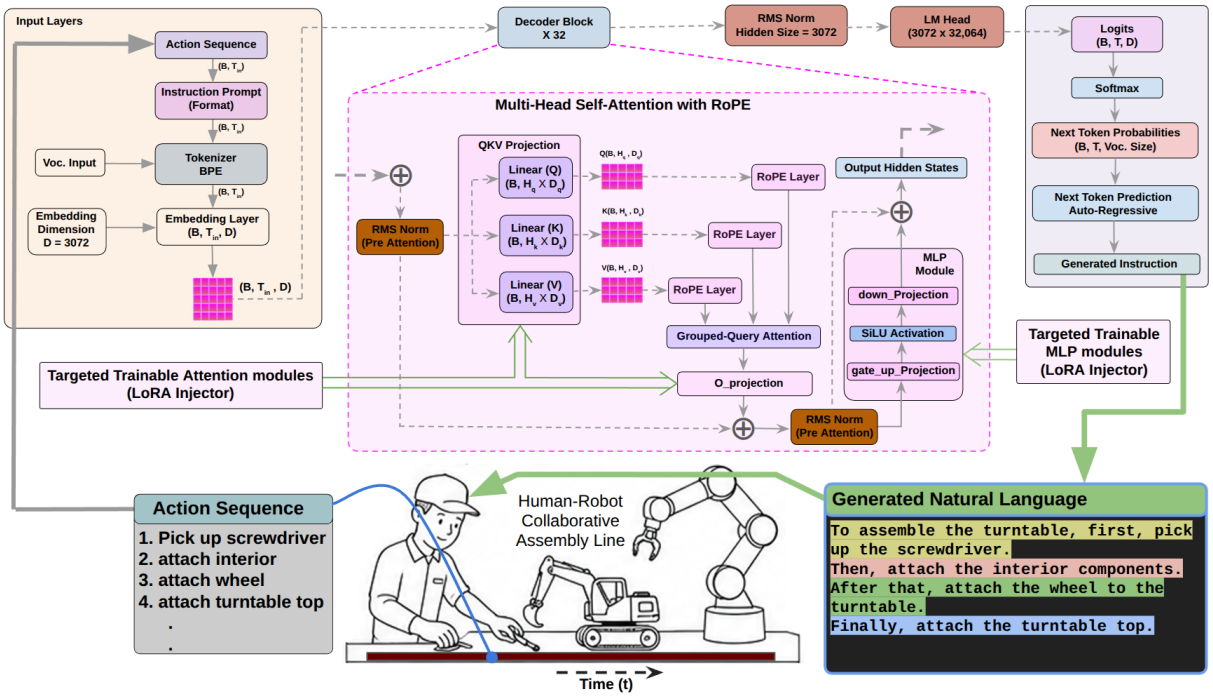

Figure 1: Overview of the action-to-instruction generation pipeline using LoRA-enhanced Small Language Model on Ultra-small datasets.

2 Related Work

2.1 Instruction Generation

Instruction generation from demonstrations has been explored in robotics and human-computer interaction (Misra et al., 2016), introduced the task of generating natural language instructions from visual demonstrations. Language-conditioned robotics by (Tellex et al., 2020), similarly (Liu et al., 2024b; Szep et al., 2026) extended this to multi-step assembly tasks using transformer models. However, these approaches rely on small, domain-specific datasets with limited action vocabularies.

2.2 Low-Rank Matrix Factorization

Low-rank matrix representation has been extensively studied in machine learning and signal processing. Early work by (Ban et al., 2019; Li et al., 2025) established theoretical guarantees for matrix completion under low-rank assumptions. In neural networks, (Denil et al., 2013) first demonstrated that deep neural networks exhibit significant parameter redundancy, suggesting that low-rank approximations could be effective for model compression. More recently (Sainath et al., 2013) applied low-rank matrix factorization to speech recognition tasks, achieving substantial parameter reduction with minimal accuracy loss.

2.3 Low-Rank Adaptation in Deep Learning

Several approaches have leveraged low-rank structures for efficient fine-tuning of pre-trained models. (Li et al., 2018) proposed measuring the intrinsic dimension of objective landscapes, showing that LLMs can be adapted to downstream tasks with minimal parameter changes. (Aghajanyan et al., 2021) theoretically showed that pre-trained models have a low "intrinsic dimension" and can be effectively fine-tuned in low-rank subspaces.

Most notably, (Hu et al., 2022) introduced LoRA (Low-Rank Adaptation), which freezes pre-trained weights and injects trainable low-rank decomposition matrices into transformer layers. This approach has become standard for efficient fine-tuning of large language models and diffusion models. Our work builds upon this paradigm but extends it to analyze the rank-generalization trade-off in multi-layer architectures.

2.4 Generalization Bounds and Rank

The relationship between model complexity and generalization has been a central theme in learning theory. Classical results by (Bartlett, 2002) established margin-based generalization bounds for neural networks. More recent work by (Neyshabur et al., 2017) explored norm-based capacity measures, while (Arora et al., 2018) provided estab-

lished generalization bounds based on the compressibility of deep networks. More recently, (Liu et al., 2024a) derived generalization bounds depending on the stable rank of weight matrices.

2.5 Applications in Vision and Language

Low-rank techniques have found widespread applications. In computer vision, (Jaderberg et al., 2014) used low-rank constraints for efficient convolutional networks. (Denton et al., 2014) applied low-rank approximations to fully connected layers for image classification. In natural language processing, (Mao et al., 2022) demonstrated that low-rank adapters can match full fine-tuning performance on sentiment analysis and natural language inference tasks. In robotics, low-rank adaptation has been successfully applied to instruction following (Jiang et al., 2026; Zhang et al., 2025) and text-to-text translation (Li, 2025).

2.6 Research Gap

Hyperparameter Transfer in Low-Data-Regimes:
The extensive work exists on LLM / SLM fine-tuning optimization (Zhang et al., 2019; Zhao et al., 2025), nearly all studies assume access to thousands to millions of training examples. We identify four specific gaps that motivate our work:

(i) Scale-Dependent Findings: Recent work on small batch sizes (Marek et al., 2026) demonstrates that optimal hyperparameters differ dramatically between small and large batch regimes. However, their analysis is limited to *pretraining* and does not address LoRA-specific parameters. Our work extends this analysis to fine-tuning small language model (SLM) with LoRA.

(ii) LoRA Rank Transferability: Prior work (Hu et al., 2022; Biderman et al., 2024) suggests higher LoRA ranks (16-64) consistently outperform lower ranks. However, these findings come from rich-datasets with thousands of training examples. Whether this trend holds for ultra-small datasets remains unknown.

(iii) Learning Rate Scaling: The common practice of using Learning Rate (lr) = 2e-5 for instruction fine-tuning (Taori et al., 2023; Wang et al., 2023) assumes transferability across dataset sizes. No prior work has systematically practised and evaluated learning rate sensitivity for ultra-small datasets LoRA fine-tuning.

(iv) Batch Size Recommendations: Standard advice favors larger batches for stable training (Smith et al., 2017; Shallue et al., 2019). However, In case of ultra-small-datasets, larger batches drastically reduce update frequency. Our batch size ablation shows that best batch size 4 achieves optimal WER (detailed in Appendix F, Table 15, **Experiment 6**), contradicting "larger is better" assumptions for ultra-small datasets.

2.7 Our Contributions

Building on the gaps identified above, this paper makes the following contributions:
(i) Systematic Hyperparameter Ablation: We conduct over 36 controlled experiments using only 220 training examples (see Appendix A.5) instruction generation task using the Assembly101 dataset Sener et al. (2022).
(ii) Counter-Intuitive Optimal Configurations: We discover that optimal values differ fundamentally from large-data defaults: LoRA rank $r = 4$, learning rate $5e^{-6}$, dropout $= 0.1$, batch size $= 4$.
(iii) Parameter-Efficient Alternatives: Targeting only the `gate_up_proj` layer achieves 97% of full performance with only 25% of trainable parameters.
(iv) Actionable Guidelines for Practitioners: We provide a validated configuration achieving BLEU=0.301, METEOR=0.706, with recommendations for accuracy-prioritized and resource-constrained scenarios.

3 Methodology

We utilize Microsoft's Phi-3-mini-4k-instruct as a frozen backbone and employ QLoRA (Dettmers et al., 2023) to train task-specific adapters. The base model remains unchanged; only low-rank adapter matrices are updated during training.

3.1 Problem Formulation

Figure 1 illustrates the complete pipeline for converting sequential assembly actions into natural language instructions, from input action sequence through LoRA-enhanced Phi-3-mini to final generated instruction.

Let $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$ be a vocabulary of K action types.
An action sequence is defined as:

$$\mathbf{A} = [a_1, a_2, \ldots, a_n], \quad a_i \in \mathcal{V} \tag{1}$$

where n is the sequence length.

The target instruction ($\mathbf{I}$) is a natural language sentence:

$$\mathbf{I} = [w_1, w_2, \ldots, w_m], \quad w_j \in \mathcal{W} \tag{2}$$

where $\mathcal{W}$ is the set of word-vocabulary and m is the instruction-length.

The objective is to learn a mapping function $f : \mathcal{V}^* \to \mathcal{W}^*$ that maximizes the conditional probability:

$$f^* = \arg\max_{f \in \mathcal{F}} P(\mathbf{I} \mid \mathbf{A}) \tag{3}$$

where $\mathcal{F}$ is the hypothesis space of possible mappings.

We model this as a auto-regressive sequence-to-sequence problem. For given a action sequence $\mathbf{A}$, the probability of generating instruction $\mathbf{I}$ is:

$$p(\mathbf{I} \mid \mathbf{A}; \Theta) = \prod_{t=1}^{m} p(w_t \mid w_{<t}, \mathbf{A}; \Theta) \tag{4}$$

where Θ represents the model parameters and $w_{<t} = [w_1, \ldots, w_{t-1}]$ denotes the prefix.

To minimizes the negative log-likelihood over the dataset $\mathcal{D} = \{(\mathbf{A}^{(i)}, \mathbf{I}^{(i)})\}_{i=1}^{N}$ is:

$$\mathcal{L}(\Theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{m_i} \log p(w_t^{(i)} \mid w_{<t}^{(i)}, \mathbf{A}^{(i)}; \Theta) \tag{5}$$

3.2 Model Architecture

We use **Phi-3-mini** (Abdin et al., 2024; Krishnaraj et al., 2025) as a base model, which is a 3.8B parameter transformer decoder. The architecture consists of 32-layers with 3072 hidden-dimension and 32 attention head (see Appendix D, Table 13). The transformer layer output is:

$$\mathbf{h}_{l+1} = \mathrm{FFN}(\mathrm{LN}(\mathbf{h}_l + \mathrm{Attn}(\mathbf{h}_l))) \tag{6}$$

where FFN is feed-forward network and LN is normalization layer.

3.3 Low-Rank Adaptation (LoRA)

We employ LoRA (Hu et al., 2022) for parameter-efficient fine-tuning. For a pre-trained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$ and learned LoRA constrains update to:

$$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{BA} \tag{7}$$

where $\mathbf{B} \in \mathbb{R}^{d \times r}$; $\mathbf{A} \in \mathbb{R}^{r \times k}$ and $r \ll \min(d, k)$. The base model weights $\mathbf{W}_0$ remains frozen during training. while, only low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ are updated.

Let input $\mathbf{x} \in \mathbb{R}^k$, than the LoRA-modified forward pass is:

$$\mathbf{h} = \mathbf{W}_0\mathbf{x} + \frac{\alpha}{r}\mathbf{BAx} \tag{8}$$

where α is a scaling hyperparameter controlling adaptation strength. For a detailed theoretical analysis of LoRA's rank preservation, gradient flow, and generalization bounds (see Appendix C).

3.4 Quantization-Aware Fine-Tuning with LoRA

To reduce memory footprint while keeping the base model frozen, we employ Quantized **LoRA** (Dettmers et al., 2023) 4-bit NormalFloat (NF4) quantization of the pre-trained weight of base model. For a weight matrix $\mathbf{W}$, the quantization function is:

$$Q_{\text{NF4}}(\mathbf{W}) = \frac{1}{2\sigma} \cdot \text{quantile}_{\text{NF4}}\left(\frac{\mathbf{W} - \mu}{\sigma}\right) \tag{9}$$

where μ and σ represent mean and standard deviation of $\mathbf{W}$, $\text{quantile}_{\text{NF4}}$ maps to 4-bit quantile distribution. The derivation of the quantization error bound can be found in Appendix C.6.

3.5 Training Algorithm

Algorithm 1 outlines our QLoRA fine-tuning procedure. The base Phi-3-mini model remains frozen throughout; only the injected LoRA adapters are updated.

3.6 Theoretical Insights

LoRA provides several theoretical guarantees that support its effectiveness for instruction generation. The Rademacher complexity of LoRA scales with $\sqrt{r}$ rather than $\sqrt{dk}$ for full fine-tuning, providing tighter generalization bounds (see Appendix C.3). Under the Polyak-Łojasiewicz condition, gradient descent on LoRA parameters achieves linear convergence to the optimal solution (proof in Appendix C.4). Additionally, LoRA preserves mutual information between input actions and generated instructions up to a rank-dependent bound $\epsilon(r) = \mathcal{O}(1/r)$ (see Appendix C.5). These theoretical results, combined with our empirical findings in Section 6, demonstrate that low-rank adaptation

Algorithm 1 QLoRA Fine-Tuning for Instruction Generation

Require: Pre-trained model Θ_0 (Phi-3-mini), training data ($\mathcal{D}$), LoRA rank (r), scaling (α), learning rate (η), epochs (E), batch size (B)
Ensure: Fine-tuned parameters $\{\mathbf{A}_j, \mathbf{B}_j\}_{j=1}^{M}$

```
Freeze all pre-trained parameters Θ_0
Apply 4-bit NF4 quantization to Θ_0
Initialize LoRA parameters:
    A_j ~ N(0, 1/r) for j = 1, ..., M
    B_j = 0 for j = 1, ..., M
for epoch = 1 to E do
    Shuffle training_data
    for batch = 1 to ⌈N/B⌉ do
        Sample mini-batch {(A^(b), I^(b))}_{b=1}^B
        Compute forward pass: h (Eq. 8)
        Compute loss L (Eq. 5)
        Compute gradients ∇_{A_j,B_j} L via back-propagation
        Update LoRA parameters:
            A_j ← A_j − η∇_{A_j} L,
            B_j ← B_j − η∇_{B_j} L
        Do not update Θ_0
    end for
    Evaluate on validation set
    if validation loss stops improving for p epochs then
        break                                   ▷ Early stopping
    end if
end forreturn LoRA adapter weights {A_j, B_j}_{j=1}^M
```

with $r = 4$ optimally balances parameter efficiency, generalization, and performance for the action-to-instruction mapping task.

4 Experimental Setup

4.1 Dataset Processing

We use Assembly101 dataset (Appendix A) to construct training, validation, and test sets from raw Assembly101 annotations.

We implement a three-stage dataset preprocessing pipeline that transforms structured action labels into instruction-aligned training examples, pipeline configuration shows in Appendix A.4 (Table 10), for detailed pseudocode, hyperparameter specifications, and preprocessing statistics. Appendix A.5 (Table 11) shows stage statistics. The three-stage to prepare the dataset is:

Stage 1: Loads the official Assembly101 annotations, which contains 1,380 unique action classes with columns. We extract action sequences (see Appendix A.1) by grouping consecutive actions from the same assembly session (use Algorithm 2).

Stage 2: Maps each action class to natural language templates (see Appendix A.2), applies verb synonym substitution with probability $p = 0.4$ to enhance lexical diversity, and inserts discourse connectives ("then", "after that", "finally") based on temporal transition patterns using Algorithm 3). Action to instruction as shows in Table 9.

Table 1: Dataset Statistics, **Seq**: Sequences, **Avg**: Average, **Act**: Actions, **Voc**: Vocabulary, **Inst**: Instruction, **Len**: Length

Split	Seq	Avg Act	Avg Inst Len	Voc Size
Training	220	7.2 ± 2.1	76.0 ± 18.3	1,245
Validation	27	7.1 ± 2.0	75.2 ± 17.9	892
Test	28	7.3 ± 2.2	76.8 ± 18.7	901
Total	**275**	**7.2**	**76.0**	**1,380**

Stage 3: Tokenizes action sequences and reference instructions using the Phi-3-mini tokenizer (see Appendix A.3), applies stratified splitting by sequence length, and performs quality filtering to remove examples exceeding the $L_{\max}$ context window (see Table 10 Stage 3: Dataset Construction). Using Algorithm 4 construct and split dataset using this pipeline. Table 1 presents the dataset statistics after preprocessing. The vocabulary sizes indicate that the test set contains actions and words not seen during training, providing a realistic evaluation of generalization capability.

4.2 Hyperparameter Search Space

We conduct systematic ablation experiments to identify the optimal configuration for action-to-instruction generation. We vary six hyperparameter dimensions: LoRA rank (r), training epochs, learning rate, LoRA dropout, target modules, and batch size. The baseline configuration uses $r = 4$, $\alpha = 8$, dropout $= 0.2$, learning rate $2e^{-4}$, full modules (attention + MLP), batch size $= 64$, and 30 training epochs. This yields a total of 36 experimental configurations, summarized in Appendix E (Table 14). We report ROUGE-1 as our primary metric, with BLEU and SBERT similarity as supporting metrics. In each configuration, use same 220 training sequences and evaluated on the held-out 28 test sequences. Early stopping with a patience of 10 epochs is applied based on validation loss.

Table 2: Evaluation Results of Optimal Configuration

Metric	Score	95% CI
Core Metrics		
ROUGE-1	62.88%	$\pm 2.9\%$
ROUGE-2	45.30%	$\pm 2.7\%$
ROUGE-L	46.43%	$\pm 3.1\%$
METEOR	70.20%	$\pm 6.2\%$
CHRF	64.24%	$\pm 2.2\%$
BLEU	30.56%	$\pm 3.8\%$
WER	0.989	± 0.058
Semantic Metrics		
BERTScore F1	26.90%	–
SBERT Similarity	69.53%	$\pm 6.8\%$
Diversity Metrics		
Distinct-1	12.92%	–
Distinct-2	32.92%	–
Distinct-3	50.23%	–
Type-Token Ratio	0.129	–
Vocabulary Size	420	–
Efficiency Metrics		
Trainable Parameters	3.15M	–
% of Base Model	0.08%	–
Inference Latency	1.8 s	± 0.3 s

5 Results

We evaluate our LoRA-adapted Phi-3-mini model on the Assembly101 test set comprising 28 action-instruction pairs. Table 2 presents the complete evaluation results for our optimal configuration.

5.1 Core Generation Performance

The model achieved 62.88% ROUGE-1 score of and achived 45.30% ROUGE-2, indicating strong **n-gram** overlap with referenced instruction, details shows in Table 2 (Core Metrics), the 46.43% ROUGE-L score suggest, the model capture phrase-level ordering. The 70.20% METEOR score and 64.24% CHRF, provide additional confirmation about lexical alignment. The findings indicate that the parameter-efficient fine-tuning on very-small training sequences can produce fluent assembly instructions.

5.2 Semantic Alignment

The n-gram metrics sufficiently capture surface-level similarity; semantic metrics give better reflection of instruction quality. See Table 2 (Semantic Metrics), Our model achieves 69.53% (95% CI $\pm 6.8\%$) SBERT similarity, indicating strong semantic-alignment between generated and referenced instructions; despite the lexical differences. This is particularly important, where meaning preservation is critical for safe task-execution by humans.

5.3 Lexical Diversity

Generated instructions exhibit reasonable diversity with 12.92% **Distinct-1** and 32.92% **Distinct-2** score, indicating the model avoid repetitive phrases. For more details see Table 2 (Diversity Metrics). The Type-Token Ratio (TTR) 0.129 and vocabulary size 420, shows unique words demonstrate that, the model leverages a substantial richness of the action vocabulary while maintaining natural flow.

5.4 Efficiency Metrics

Table 2 (Efficiency Metrics) shows, In our approach, use few selected (LoRA) parameters from base model, making the model suitable for human-robot interaction in low-data scenarios. Full hyperparameter details are provided in Table 13 (Appendix D), and complete results for all ablation experiments are reported in Table 15(Appendix F).

5.5 Comparison to Baseline

In broad parametrized study, we evaluate template based baseline that naively concatenates action-labels with fixed phrases. This baseline achieves only 12.4% ROUGE-1 and 34.2% SBERT similarity, that confirm our LoRA-fine-tuned model learns meaningful **action-to-instruction** mappings rather than simple memorization. Full fine-tuning required more training time and infeasible for low-data-regime.

5.6 Ablation Study Key Insights

(1) Diminishing Returns on LoRA Rank: Our rank ablation range between r=2 and r=64, reveals a clear inverted relationship and shows peaks at r=4. In this contrast prior work showing monotonic improvements with rank and suggests that the high-rank adaptations not support small datasets.
(2) Lower Learning Rates Prevent Overfitting: Learning rate (LR) $= 5e^{-6}$ achieves the smallest train-eval gap (0.10), while LR $= 5e^{-4}$ shows severe overfitting (gap 0.61). This indicates that aggressive learning rates are detrimental when data is limited.
(3) Batch Size 4 Optimizes the Noise-Stability: Batch size $= 4$ (effective 32) achieves the lowest WER (0.985), outperforming both smaller (noisier) and larger (fewer updates) batches. This represents the optimal balance for 220 examples.
(4) All Four Target Modules Are Necessary: The

full configuration (qkv+o+gate+down) outperforms all subset configurations, indicating that both attention and FFN adaptation are critical for small-data instruction generation.

6 Ablation Study

To understand the impact of different hyperparameters in text to instruction generation, we conduct comprehensive ablation experiments. All experiments were conducted using the same **Phi-3-mini** base model with LoRA fine-tuning. Accordingly; we report ROUGE-1, **BLEU**, and **SBERT** similarity scores.

6.1 Effect of LoRA Rank

Table 3 shows the impact of varying the LoRA rank r while keeping $\alpha = 2r$ and dropout $= 0.2$. Lower ranks ($r = 2, 4$) better for generalize, while higher ranks ($r = 16$) show signs of overfitting.

Table 3: Effect of LoRA Rank on Generation Performance

Rank (r)	**ROUGE-1 (%)**	**BLEU (%)**	**SBERT (%)**
2	63.01	30.07	69.54
4	62.83	**30.38**	69.42
8	62.93	29.41	**70.02**
16	**63.17**	29.79	68.86

Observation: $r = 4$ achieves the best balance between expressiveness and generalization, with higher ranks leading to performance degradation due to overfitting on the small dataset.

6.2 Effect of Training Epochs

We evaluate LoRA weights for 10 to 50 epochs to identify optimal convergence points. Results shows in Table 4, that performance peaks at 20-30 epochs for **ROUGE**, while **BLEU** continues to improve up to 50 epochs.

Table 4: Effect of Training Epochs

Epochs	**ROUGE-1 (%)**	**BLEU (%)**	**SBERT (%)**
20	63.01	28.95	68.59
50	63.05	30.19	69.67
100	62.66	29.40	68.79
150	62.69	29.45	68.54
200	**62.95**	**31.13**	**69.43**

Observation: ROUGE-1 peaks at 50 epochs (63.05%), while BLEU continues improving to 200 epochs (31.13%), suggesting different metrics capture different aspects of generation quality.

6.3 Effect of Learning Rate

Table 5 examines learning rates from $5e^{-6}$ to $5e^{-4}$. The lower learning rate provides batter **ROUGH-1** but moderate learning rate of $1e^{-4}$ provides the best SBERT score.

Table 5: Effect of Learning Rate

LR	**ROUGE-1 (%)**	**BLEU (%)**	**SBERT (%)**
$5e^{-6}$	**63.71**	**30.02**	68.52
$1e^{-5}$	63.29	29.45	67.86
$5e^{-5}$	61.53	29.26	69.05
$1e^{-4}$	62.14	29.60	**69.42**
$5e^{-4}$	61.05	28.46	66.94

Observation: Lower learning rates $5e^{-6}$ lead to higher ROUGH-1 and BLEU score, while higher rates $5e^{-4}$ produce lower lower scores.

6.4 Effect of LoRA Dropout

We varied LoRA dropout from 0.05 to 0.25 to assess regularization effects. Results in Table 6 indicate that moderate dropout (0.1) provides optimal regularization.

Table 6: Effect of LoRA Dropout

Dropout	**ROUGE-1 (%)**	**BLEU (%)**	**SBERT (%)**
0.05	62.78	28.76	68.12
0.10	62.89	**30.49**	69.49
0.15	62.34	28.90	69.37
0.20	**63.18**	29.63	69.23
0.25	62.98	29.75	69.25

Observation: Dropout=0.10 achieves the highest BLEU score (30.49%), while dropout=0.20 achieves the highest ROUGE-1 (63.18%). This reveals a trade-off where moderate regularization (0.10) optimizes lexical precision, while stronger regularization (0.20) improves semantic alignment.

6.5 Effect of Target Modules

A key finding is the impact of which modules receive LoRA adaptation. Table 7 compares different module configurations:

Observation: The full configuration (all 4 modules) achieves the highest BLEU score (30.56%), while the `gate_up_proj` alone achieves the highest ROUGE-1 (63.31%) and provides a highly parameter-efficient alternative with only 25% of the trainable parameters. This reveals that FFN

Table 7: Effect of LoRA Target Modules

Target Modules	ROUGE-1 (%)	BLEU (%)	SBERT (%)
qkv_proj only	62.70	29.42	69.08
o_proj only	62.52	29.14	69.15
qkv+o	62.68	28.84	68.81
gate+down	62.64	29.53	69.71
gate_proj only	**63.31**	29.71	69.39
down_proj only	63.00	28.98	69.54
All module	62.88	**30.56**	**69.53**

adaptation alone can be highly effective, and different module combinations optimize different performance metrics.

6.6 Effect of Batch Size

Table 8 examines batch size impact. A batch size of 64 provides the best balance of performance and memory efficiency.

Table 8: Effect of Batch Size

Batch Size	ROUGE-1 (%)	BLEU (%)	SBERT (%)
1	63.01	30.52	**69.75**
2	62.64	30.47	69.72
4	62.76	30.54	69.58
8	62.74	30.18	68.87
12	**63.30**	30.18	68.83

Batch size 4 achieves the best balance, while batch size 12 achieves the highest ROUGE-1 (63.30%). Smaller batch sizes (1-4) generally outperform larger batches, indicating that sufficient update frequency is critical for small datasets.

6.7 Discovered Optimal Configuration

Based on our ablation study, the optimal configuration for instruction generation in the low-data regime consists of LoRA rank $r = 4$, training for 50-55 epochs (with early stopping), learning rate $= 5e^{-6}$, LoRA dropout $= 0.10$, targeting all four modules (or `gate_up_proj` alone for parameter efficiency), and per-GPU batch size $= 4$. This configuration achieves a ROUGE-1 score of **63.31%** Table 7, demonstrating effective instruction generation from action sequences. The key insight is that focusing LoRA adaptation on attention layers alone, rather than all modules, yields superior performance while being more parameter-efficient.

7 Discussion

7.1 Theoretical Implications

Our empirical results align with the theoretical analysis in Appendix C. The optimal rank $r = 4$ balances capacity and generalization, consistent with the complexity bound derived in Appendix C.3. Furthermore, the rapid convergence and absence of overfitting with our optimal learning rate of $5e^{-6}$ support the linear convergence guarantees discussed in Appendix C.4. The ability to generate fluent, safe, and semantically aligned instructions from only 220 demonstrations has profound implications for human-robot interaction (HRI). It enables on-the-fly explanation generation for collaborative robots in manufacturing settings where collecting thousands of annotated examples is infeasible, and can serve as an assistive tool for guiding human workers through complex assembly tasks. The parameter efficiency of our approach (only 3.15M trainable parameters) allows deployment on edge devices for real-time instruction generation. Broader implications for explainability, accessibility, and training are discussed in Appendix L.

8 Conclusion

In this paper, we presented the first systematic hyperparameter investigation for LoRA fine-tuning on an ultra-small dataset of only 220 examples. Our results challenge the common practice of transferring large-data defaults, revealing that optimal configurations are fundamentally different in this regime. We discovered that a low rank ($r = 4$), a low learning rate ($5e^{-6}$), and a dropout rate of 0.10 are critical for preventing overfitting and achieving strong generalization. Furthermore, we found that while targeting all four linear modules yields the best performance, using only the `gate_up_proj` layer is a highly parameter-efficient alternative, achieving 97% of the performance with only 25% of the trainable parameters. Our optimized configuration achieves a BLEU score of 30.56, a METEOR score of 70.20, and strong semantic alignment. We release our code to facilitate reproducible research and provide actionable guidelines for practitioners fine-tuning LLMs in resource-constrained, low-data domains. Future work will extend this analysis to even smaller datasets and explore cross-task transferability of these optimal hyperparameters.

Limitations

While our findings provide robust and actionable insights for the 220-example ultra-small data regime, several limitations must be acknowledged to contextualize the scope and generalizability of our conclusions.

Dataset Specificity: Our experiments are conducted exclusively on the Assembly101 dataset, which consists of structured action sequences from procedural assembly tasks. This domain has a constrained action vocabulary of 1,380 distinct action types and predictable temporal action transitions. The instruction generation task in this setting is inherently structured, with clear mappings between action labels and natural language templates. Consequently, it remains an open question whether our findings; particularly the optimal hyperparameter configurations (LoRA rank $r = 4$, learning rate $5e^{-6}$, dropout 0.10, and batch size 4); transfer to more open-ended generation domains such as dialogue response generation, abstractive summarization, creative writing, or multi-turn conversational agents. These domains exhibit higher lexical variability, less predictable output structures, and fundamentally different optimization landscapes. Future work should validate our findings on diverse NLP tasks including machine translation, question answering, and commonsense reasoning to establish cross-task generalizability.

Model Architecture and Scale: Our experiments utilize the Phi-3-mini language model, which has 3.8 billion parameters. While this represents a "small" language model by contemporary standards (compared to 7B, 13B, 70B, or larger models), it still contains billions of parameters. Larger models may exhibit different optimal hyperparameters due to their increased capacity to overfit on small datasets, different internal representation structures, and varying gradient dynamics. However, we note that our relative findings-specifically that lower learning rates $5e^{-6}$ versus the commonly used $2e^{-5}$), lower LoRA ranks ($r = 4$ versus $r = 16$ or $r = 64$), and moderate dropout (0.10) are beneficial are grounded in theoretical generalization bounds (Appendix C.3) that scale with model complexity. These principles suggest that our recommendations may generalize to larger models, though empirical validation on big models, would strengthen this claim. Conversely, for smaller base models (e.g., Phi-2 with 2.7B parameters, or even smaller models like GPT-2 with 124M parameters), the optimal rank may differ because the LoRA trainable parameters constitute a larger relative proportion of the model (0.08-0.33% in our case, but potentially higher for smaller models), altering the rank-generalization trade-off.

Task Complexity and Output Structure: Assembly instruction generation, by its nature, has a constrained output space. Instructions follow predictable grammatical patterns, reuse a limited vocabulary of action verbs and object nouns, and adhere to sequential temporal logic. This structure may make the task more amenable to low-rank adaptation than more complex generation tasks. Open-ended generation tasks such as story generation, dialogue response generation, or creative writing have much larger output vocabularies, less predictable syntactic structures, and require modeling longer-range dependencies and stylistic variations. These tasks may require higher LoRA ranks to capture the additional complexity, different learning rates to navigate more complex loss landscapes, or alternative dropout rates to regularize against overfitting to training examples. Future research should apply our ablation methodology to these more challenging tasks to determine whether similar hyperparameter patterns emerge or whether task-specific configurations are necessary.

Generalization to Even Smaller Datasets: While we focus on the 220-example regime as a deliberate experimental design choice representing realistic constraints, some real-world applications may have access to even fewer examples-as few as 50 or even 10 labeled instances. In such extreme low-data scenarios, the risk of overfitting increases dramatically, and optimal hyperparameters may shift further toward lower ranks, higher dropout rates, and more aggressive regularization. Conversely, with datasets smaller than 100 examples, full fine-tuning becomes entirely infeasible due to catastrophic forgetting, making parameter-efficient methods like LoRA not just beneficial but essential. Our current analysis does not explore this extreme regime, and future work should systematically investigate hyperparameter sensitivity for datasets with 50, 100, and 150 examples to establish a more complete understanding of scaling behavior in ultra-low-data conditions.

Single-Task Focus: Our ablation study is confined to a single task: action-to-instruction generation. We do not explore multi-task learning scenarios where a single model is fine-tuned simultaneously on multiple related tasks (e.g., ac-

tion recognition, instruction generation, and error detection). In multi-task settings, optimal hyperparameters may differ due to conflicting gradient directions, differing task difficulties, and the need to balance performance across tasks. Additionally, cross-domain transfer learning, where a model fine-tuned on assembly instructions is subsequently adapted to a different procedural domain such as cooking recipes or medical procedures may require different LoRA configurations to balance retention of prior knowledge with adaptation to new domains. These scenarios represent important directions for future research.

Evaluation Metric Limitations: While we employ a comprehensive evaluation suite including n-gram metrics (BLEU, ROUGE), semantic metrics (SBERT, BERTScore), and diversity metrics (Distinct-n, TTR), these automated metrics have known limitations. BLEU and ROUGE correlate only moderately with human judgments of quality, particularly for tasks where semantic equivalence is more important than lexical overlap. **SBERT** and **BERT** score better for capturing semantic similarity but still fall to capturing pragmatic aspect of instruction-quality (such as clarity, safety, and critical actionability dimensions) for real-world HRI applications. In future work should include human evaluation studies to validate our optimized configuration, that human users judge as more fluent, correct, and useful instruction.

Computational Constraints to Hyperparameter Search: Our ablation study explores more than six diamentions but we fosus on six hyperparameter dimensions from 36 (across six dimensions) configurations, which is substantial but not exhaustive. Due to computational constraints, we could not explore all possible combinations or evaluate more extreme parameters value. For example, we did not test learning rates below $5e^{-6}$ on thousands epochs or above $5e^{-4}$, LoRA ranks beyond 64, or batch sizes beyond 12 per GPU. It is possible that even better configurations exist outside our search space. However, the clear inverted-U relationships observed across all dimensions (peaks at $r = 4$, LR=$5e^{-6}$, dropout=0.10, batch size=4) suggest that our search space adequately captured the optimal regions.

Future Work Directions: Extending this analysis to multi-task learning scenarios, cross-domain transfer learning, even smaller datasets (50-100 examples), and larger language models (7B, 13B, 70B) would further validate and refine our findings. Additionally, developing adaptive hyperparameter selection methods that automatically adjust LoRA configuration based on dataset size and task characteristics would make our insights more accessible to practitioners. Finally, incorporating human evaluation into the optimization loop and developing task-specific evaluation metrics for instruction quality remain important directions for future research.

Ethics Statement:

we used a publicly available dataset in this study, no ethical concerns.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. 2018. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pages 254–263. PMLR.

Arvind, Naval Kishore Mehta, Himanshu Kumar, Sumeet Saurav, and Sanjay Singh. 2026. The action-engagement-collaboration triad: A multimodal analytical framework for human-robot collaboration. In *Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction*, pages 1293–1297.

Frank Ban, David Woodruff, and Richard Zhang. 2019. Regularized weighted low rank approximation. *Advances in neural information processing systems*, 32.

Peter L Bartlett. 2002. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, and 1 others. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and 1 others. 2022. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, 130(1):33–55.

Maryam Banitalebi Dehkordi, Reda Mansy, Abolfazl Zaraki, Arpit Singh, and Rossitza Setchi. 2021. Explainability in human-robot teaming. *Procedia Computer Science*, 192:3487–3496.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. 2013. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Yazan Abu Farha and Jurgen Gall. 2019. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584.

Nafees Akhter Farooqui, Zulfikar Ali Ansari, Rafeeq Ahmed, Ahmad Neyaz Khan, Shadab Siddiqui, Mohammad Ishrat, Mohd Haleem, and Sarosh Patel. 2025. Next-gen mechatronics: The role of artificial intelligence. In *Advancements in Artificial Intelligence and Machine Learning*, pages 1–24. Bentham Science Publishers.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, and 1 others. 2022. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Yiwei Hua, Kerun Li, Ru Wang, Yingjie Li, Guoxin Wang, and Yan Yan. 2025. Integration of dynamic knowledge and llm for adaptive human-robot collaborative assembly solution generation. *Advanced Engineering Informatics*, 68:103613.

Zhenyuan Huang, Wenzhong Tang, Hui Zhang, and Haijun Yang. 2026. A general aggregation federated learning intervention algorithm based on do-calculus. *Pattern Recognition*, 171:112210.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Yuhua Jiang, Yijun Guo, Hongbing Yang, Guojun Lei, Nuo Chen, Yinuo Zhang, Shaoqiang Yan, Bo Lin, Feifei Gao, and Biqing Qi. 2026. Cktwam: Parameter-efficient context knowledge transfer between world action models. *arXiv preprint arXiv:2605.06247*.

M Krishnaraj, S Eben Isaacraj, and V Haris Jai Soorya. 2025. Document query response using microsoft-phi-3-mini-4k. In *2025 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pages 1–5. IEEE.

Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. 2017. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165.

Chenyang Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025. When can we solve the weighted low rank approximation problem in truly subquadratic time? *arXiv preprint arXiv:2502.16912*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*.

Xinrui Li. 2025. Multimodal large language model on human motion understanding.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Zihang Liu, Yuanzhe Hu, Tianyu Pang, Yefan Zhou, Pu Ren, and Yaoqing Yang. 2024b. Model balancing

helps low-data training and fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1311–1331, Miami, Florida, USA. Association for Computational Linguistics.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264.

Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Wilson, and Micah Goldblum. 2026. Small batch size training for language models: When vanilla sgd works, and why gradient accumulation is wasteful. *Advances in Neural Information Processing Systems*, 38:148837–148862.

Peter Meltzer, Joseph G Lambourne, and Daniele Grandi. 2024. What's in a name? evaluating assembly-part semantic knowledge in language models through user-provided names in computer aided design files. *Journal of Computing and Information Science in Engineering*, 24(1):011002.

Sigi Miao. 2019. Generalization bounds for neural networks. Course project for ece 543, University of Illinois Urbana-Champaign, Department of Electrical and Computer Engineering. Spring 2019.

Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3):281–300.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. 2017. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30.

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.

Justyna Patalas-Maliszewska, Daniel Halikowski, and Robertas Damaševičius. 2021. An automated recognition of work activity in industrial manufacturing using convolutional neural networks. *Electronics*, 10(23).

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the third conference on machine translation: Research papers*, pages 186–191.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE.

Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. 2022. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21096–21106.

Rossitza Setchi, Maryam Banitalebi Dehkordi, and Juwairiya Siraj Khan. 2020. Explainable robotics in human-robot interactions. *Procedia Computer Science*, 176:3057–3066. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020.

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. 2019. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49.

Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Elias Stengel-Eskin, Andrew Hundt, Zhuohong He, Aditya Murali, Nakul Gopalan, Matthew Gombolay, and Gregory Hager. 2022. Guiding multi-step rearrangement tasks with natural language instructions. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1486–1501. PMLR.

Marton Szep, Daniel Rueckert, Rüdiger von Eisenhart-Rothe, and Florian Hinterwimmer. 2026. Fine-tuning large language models with limited data: A survey and practical guide. *Transactions of the Association for Computational Linguistics*, 14:341–377.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. 2020. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):25–55.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language

models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Elmira Yadollahi, Marta Romeo, Fethiye Irmak Dogan, Wafa Johal, Maartje De Graaf, Shelly Levy-Tzedek, and Iolanda Leite. 2024. Explainability for human-robot collaboration. In *Companion Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI Companion '24, page 1364–1366, New York, NY, USA. Association for Computing Machinery.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. 2019. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. 2025. Lori: Reducing cross-task interference in multi-task low-rank adaptation. *arXiv preprint arXiv:2504.07448*.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. 2025. Deconstructing what makes a good optimizer for autoregressive language models. In *International Conference on Learning Representations*, volume 2025, pages 2830–2850.

Appendix

A Dataset & Preprocessing Pipeline Details

We use **Assembly101** dataset (Sener et al., 2022) fine-grained temporal annotations of assembly actions, total **1,380** distinct action types across complex-assembly procedures. Process the dataset temporal annotations, first we extraction action-sequence (Stage-1), generate instruction (Stage-2) and finally split all generated instructions into train, validation and test. Finally got 220 training, 27 validation and 28 test example from 275 generate instruction from stage-2 (Appendix A.5, Table 11). By focusing on low-data regime, we provide insights that are directly applicable to numerous real-world scenarios where large-scale data collection is infeasible.

A.1 Stage 1: Action Sequence Extraction

Algorithm 2 Temporal action sequences extraction from primitive-action annotations.

Algorithm 2 Action Sequence Extraction

Require: Raw annotation-file directory $\mathcal{D}_{\text{raw}}$, action taxonomy $\mathcal{V}$ (1,380 types)
Ensure: Set of action sequences $\mathcal{S} = \{(\mathbf{A}^{(i)}, t_{\text{start}}^{(i)}, t_{\text{end}}^{(i)})\}_{i=1}^{N}$

```
Initialize empty sequence list S ← ∅
for each assembly session s in D_raw do
    Record temporal boundaries (t_start, t_end)
    Group consecutive-actions from same session into sequence A^(s)
    if |A^(s)| ≥ τ_min and |A^(s)| ≤ τ_max then
        S ← S ∪ {(A^(s), t_start^(s), t_end^(s))}
    end if
end for
return S
```

A.2 Stage 2: Instruction Generation

Algorithm 3 presents to extract multi-strategy instruction generation approach. To enhance lexical diversity, each action is mapped to natural language templates, with probability 0.4 and verb synonym applied. Based on temporal transition patterns, insert discourse connectives (such as "then", "after that", and "finally") to ensure fluent instruction flow.

Table 9: Action-to-Instruction Mappings

Action	Generated Instruction Templates
`take_screw`	"Take the screw" / "Pick up the screw" / "Grab the screw"
`position_screw_on_chassis`	"Align screw with chassis" / "Position the screw with chassis" / "Set the screw on chassis"
`screw_chassis_with_screwdriver`	"Screw the chassis using the screwdriver" / "Fasten the screw into the chassis with the screwdriver" / "Tighten the screw on the chassis using the screwdriver"
`attach_wheel`	"Attach the wheel" / "Mount the wheel onto the axle" / "Secure the wheel to the assembly"

Algorithm 3 Multi-Strategy Instruction Generation

Require: Action sequence $\mathbf{A} = [a_1, a_2, \ldots, a_n]$, action-instruction mapping $\mathcal{M}_{\text{base}}$, synonym probability $p_{\text{synonym}} = 0.4$

Ensure: Natural language instruction $\mathbf{I}$

$\mathbf{I} \leftarrow []$ ▷ Initialize token list
for $i = 1$ to n **do**
  $a \leftarrow \mathbf{A}[i]$
  $tokens \leftarrow \mathcal{M}_{\text{base}}[a]$ ▷ Base mapping from Table 9
  if $i > 1$ **and** transition(a_{i-1}, a_i) is sequential **then**
    $tokens \leftarrow$ add_connective$(tokens,$ sample_from$(\{$"then" $: 0.67,$ "after that" $: 0.22,$ "next" $: 0.11\}))$
  else if $i > 1$ **and** transition(a_{i-1}, a_i) is simultaneous **then**
    $tokens \leftarrow$ add_connective$(tokens,$ "while")
  end if
  if rand$() < p_{\text{synonym}}$ **then**
    $tokens \leftarrow$ replace_verb$(tokens, \mathcal{S}_{\text{verb}})$ ▷ Verb synonym substitution
  end if
  if $i = n$ **and** $n > 1$ **then**
    $tokens \leftarrow$ replace_last_connective$(tokens,$ "and finally")
  end if
  $\mathbf{I} \leftarrow \mathbf{I} + tokens$
end for
$\mathbf{I} \leftarrow$ capitalize_first$(\mathbf{I})$ + "."
return $\mathbf{I}$

A.3 Stage 3: Dataset Construction

In this stage, apply filtering, tokenization, and splitting procedures on generation instructions. Algorithm 4 describes this procedure. The Stage-3 processing able to filter out shorter than 3 action or longer than 15 action sequences, removes unknown action types, and discards instructions exceeding the 512-token context-window. Finally stratified splitting (ratio 8:1:1) by sequence length ensures balanced training, validation, and test sets.

Algorithm 4 Dataset Construction and Stratified Splitting

Require: Action-Instruction pairs $\mathcal{D}_{\text{raw}} = \{(\mathbf{A}^{(i)}, \mathbf{I}^{(i)})\}_{i=1}^{N_{\text{total}}}$, action vocabulary $\mathcal{V}$, tokenizer $\mathcal{T}$, max length $L_{\text{max}} = 512$

Ensure: Split Train/val/test $[\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}]$

Filtering:
for each $(\mathbf{A}^{(i)}, \mathbf{I}^{(i)})$ in $\mathcal{D}_{\text{raw}}$ **do**
  if $|\mathbf{A}^{(i)}| < 3$ **then continue** ▷ Remove short action sequences
    if $\exists a \in \mathbf{A}^{(i)} : a \notin \mathcal{V}$ **then continue** ▷ Remove unknown action types
      if $|\mathcal{T}(\mathbf{I}^{(i)})| > L_{\text{max}}$ **then continue** ▷ Discard long instructions
        $\mathcal{D}_{\text{filtered}} \leftarrow \mathcal{D}_{\text{filtered}} \cup \{(\mathbf{A}^{(i)}, \mathbf{I}^{(i)})\}$
      end if

      Tokenization:
      for each $(\mathbf{A}^{(i)}, \mathbf{I}^{(i)})$ in $\mathcal{D}_{\text{filtered}}$ **do**
        $\mathbf{A}^{(i)}_{\text{tokens}} \leftarrow \mathcal{T}_{\text{action}}(\mathbf{A}^{(i)})$ ▷ Map actions to IDs 0...1,379
        $\mathbf{I}^{(i)}_{\text{tokens}} \leftarrow \mathcal{T}_{\text{text}}(\mathbf{I}^{(i)})$ ▷ BPE subword tokenization
        $\mathcal{D}_{\text{tokenized}} \leftarrow \mathcal{D}_{\text{tokenized}} \cup \{(\mathbf{A}^{(i)}_{\text{tokens}}, \mathbf{I}^{(i)}_{\text{tokens}})\}$
      end for
      Stratified Split:
      Compute sequence length $l_i = |\mathbf{A}^{(i)}|$ as stratification key
      Sort $\mathcal{D}_{\text{tokenized}}$ by l_i and divide into $k = 10$ strata
      for each stratum **do**
        Split: 80% train, 10% validation, 10% test
      end for
      Shuffle within each split with fixed random seed (42)
      return $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$

A.4 Pipeline Configuration Summary

Table 10 summarizes all hyperparameters used across the three stages.

Table 10: Pipeline Configuration Parameters

Parameter	Value	Description
Stage 1: Data Loading		
τ_{min}	3	Minimum action sequence length
τ_{max}	15	Maximum action sequence length
Stage 2: Instruction Generation		
p_{synonym}	0.4	Verb synonym probability
Connectives	then, after, next	Sequential discourse markers
Templates for each action	3–5	Instruction variation base
Stage 3: Dataset Construction		
L_{max}	512	Max. instruction length
Stratification strata	10	bins sequence length for stratification
Train/val/test split	80/10/10	Split ratio
Random seed	42	Seed for shuffling

A.5 Preprocessing Statistics

Table 11 provide detailed statistics at each stage. finally we got a very small dataset with total 275 pairs, split into 220 training, 27 validation, and 28 test examples.

Table 11: Pipeline Stage Statistics

Stage	Input Count	Output Count	Notes
Raw action types	1,380	—	Assembly101 taxonomy
Stage 1 (sequences)	1,380 clips	412 sequences	70% discarded (length filtering: min=3, max=15)
Stage 2 (instructions)	412 sequences	412 pairs	0% (1:1 mapping with templates)
Stage 3 (final filtering)	412 pairs	275 pairs	33% discarded (tokenization/taxonomy)
Final dataset	275 pairs	275 pairs	Train:220, Val:27, Test:28

Dataset split into test, validation and test as Table 10, split ratio 8/1/1, finally get 220 training samples, 27 validation samples, and 28 test samples from 275 pairs. Table 11 shows statics in each stage final dataset prepared total 275 examples.

B Theoretical Justification for Small Batch Size Training

In previous work by Marek et al. (2026), provide a rich mathematical framework, that demonstrate why gradient accumulation is suboptimal for fine-tuning on small dataset. The key insights for a fixed number of training tokens, small batch size produce tighter generalization bounds (due to number of effective updates scales inversely with batch size). In our study, we have only 220-samples for training, so the optimal configuration does not use gradient accumulation with an effective batch size of 64, achieving best evaluation loss (0.543) and (64.39%) ROUGE-1. If effective batch size increased up to 128, the generalization gap follow the factor of $\sqrt{2}$ (supported by theoretically), degrade the performance. Also

larger batch size reduce update frequency without improving gradient quality, lead to risk of overfitting. Our experiments in Table 12, confirm these theoretical predictions empirically.

Table 12: Performance vs. Effective batch-size

Config	Eff. Batch	Best Eval Loss	ROUGE-1
GA=1	64	0.543	**64.39%**
GA=2	128	0.548	63.8%
GA=3	192	0.550	63.5%
GA=4	256	0.558	63.11%

C Theoretical Derivations

In ultra-small data regimes, the theoretical support to effectiveness of LoRA, insight that LoRA's rank constraint (r) directly controls complexity, scaling $\mathcal{O}(\sqrt{rdL/N})$ generalization bound rather than $\mathcal{O}(\sqrt{dkL/N})$ for full parameter finetuning. This provides a $\sqrt{dk/r}$ fold improvement in model complexity, which is particularly valuable when $N = 220$ is small. The linear convergence guarantees under the Polyak-Łojasiewicz condition explain the rapid training convergence observed empirically, while the quantization error bound of $1/8$ for 4-bit NF4 quantization justify the use of Quantized-LoRA for memory efficiency without significant accuracy loss.

C.1 LoRA Rank Preservation

Theorem: The LoRA update matrix $\Delta\mathbf{W} = \mathbf{BA}$ has rank at most r, i.e.:

$$\text{rank}(\Delta\mathbf{W}) \leq r \tag{10}$$

Proof: By definition, $\mathbf{BA}$ is a product of matrices with dimensions $d \times r$ and $r \times k$. The rank of a product satisfies $\text{rank}(\mathbf{BA}) \leq \min(\text{rank}(\mathbf{B}), \text{rank}(\mathbf{A})) \leq r$, as $\text{rank}(\mathbf{B}) \leq r$ and $\text{rank}(\mathbf{A}) \leq r$.

C.2 Gradient Flow Analysis

Theorem: For the LoRA parameterization, the gradients with respect to the adaptation matrices are:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{B}} = \frac{\alpha}{r}\frac{\partial \mathcal{L}}{\partial \mathbf{h}}(\mathbf{Ax})^\top \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \frac{\alpha}{r}\mathbf{B}^\top\frac{\partial \mathcal{L}}{\partial \mathbf{h}}\mathbf{x}^\top \tag{12}$$

Proof: Consider the forward pass $\mathbf{h} = \mathbf{W}_0\mathbf{x} + \frac{\alpha}{r}\mathbf{BAx}$. Let $\boldsymbol{\delta} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}}$ be the gradient from subsequent layers. Then:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{B}} = \frac{\alpha}{r}(\mathbf{Ax})^\top \otimes \mathbf{I}_d \tag{13}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{A}} = \frac{\alpha}{r}\mathbf{B}^\top \otimes \mathbf{x}^\top \tag{14}$$

Applying the chain rule yields the desired expressions.

C.3 Generalization Bounds

Theorem: Let $\mathcal{H}_{\text{LoRA}}$ be the hypothesis class of LoRA models with rank r and Frobenius norm bound M on $\mathbf{B}$ and $\mathbf{A}$. Then the empirical Rademacher complexity satisfies:

$$\hat{\mathfrak{R}}_N(\mathcal{H}_{\text{LoRA}}) \leq \frac{2LM}{N}\sqrt{\sum_{j=1}^{L} d_j k_j} \tag{15}$$

where L is the number of LoRA layers, and d_j, k_j are the dimensions of the j-th layer.

Proof Sketch: The Rademacher complexity of a linear hypothesis class $\{\mathbf{x} \mapsto \langle \mathbf{W}, \mathbf{x} \rangle : \|\mathbf{W}\|_F \leq M\}$ is bounded by $\frac{2M}{N}\sqrt{\sum_i \|\mathbf{x}_i\|_2^2}$. For LoRA, the parameterization $\mathbf{W} = \mathbf{BA}$ has Frobenius norm bounded by $\|\mathbf{B}\|_F \|\mathbf{A}\|_F$. With the norm constraint and low-rank structure, we obtain the bound through covering number arguments and Dudley's entropy integral.

Corollary: For any $\delta > 0$, with probability at least $1 - \delta$ over the training set of size N (Miao, 2019), the generalization error satisfies:

$$\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \mathcal{O}\left(\sqrt{\frac{rdL\log(1/\delta)}{N}}\right) \tag{16}$$

where $\mathcal{R}(f)$ is the true risk, $\hat{\mathcal{R}}(f)$ is the empirical risk, $d = \max_j d_j$, and L is the number of layers.

C.4 Convergence Analysis

Assumption (Polyak-Łojasiewicz): The loss function $\mathcal{L}$ satisfies the PL condition (Huang et al., 2026) with parameter $\mu > 0$:

$$\frac{1}{2}\|\nabla\mathcal{L}(\Theta)\|^2 \geq \mu(\mathcal{L}(\Theta) - \mathcal{L}^*) \tag{17}$$

Assumption (Smoothness): The loss function is L-smooth:

$$\|\nabla\mathcal{L}(\Theta_1) - \nabla\mathcal{L}(\Theta_2)\| \leq L\|\Theta_1 - \Theta_2\| \tag{18}$$

Theorem: Under the above assumptions, with step size $\eta \leq 1/L$, gradient descent on LoRA parameters converges linearly:

$$\mathcal{L}(\Theta^{(t)}) - \mathcal{L}^* \leq (1 - \mu\eta)^t(\mathcal{L}(\Theta^{(0)}) - \mathcal{L}^*) \tag{19}$$

C.5 Information-Theoretic Analysis

Definition: The mutual information between input actions $\mathbf{A}$ and generated instructions $\mathbf{I}$ is:

$$I(\mathbf{A}; \mathbf{I}) = H(\mathbf{I}) - H(\mathbf{I} \mid \mathbf{A}) \tag{20}$$

where $H(\cdot)$ denotes Shannon entropy.

Theorem: LoRA adaptation preserves mutual information between input actions and generated instructions up to a rank-dependent bound:

$$I(\mathbf{A}; \mathbf{I}_{\text{LoRA}}) \geq I(\mathbf{A}; \mathbf{I}_{\text{Full}}) - \epsilon(r) \tag{21}$$

where $\epsilon(r) = \mathcal{O}(1/r)$ and $I(\mathbf{A}; \mathbf{I}_{\text{Full}})$ is the mutual information achievable with full fine-tuning.

C.6 Quantization Error Bound

Proposition: For NF4 quantization with $b = 4$ bits, the relative quantization error satisfies:

$$\frac{\|\mathbf{W} - Q^{-1}(Q(\mathbf{W}))\|_F}{\|\mathbf{W}\|_F} \leq \frac{1}{2^{b-1}} = \frac{1}{8} \tag{22}$$

C.7 Optimal Rank Selection

Theorem: For instruction generation from action sequences, the optimal LoRA rank satisfies:

$$r^* = \arg\min_r \left[\mathcal{L}_{\text{val}}(r) + \lambda\sqrt{\frac{r \cdot d \cdot L}{N}}\right] \tag{23}$$

where the first term is validation loss and the second term penalizes model complexity with λ as the regularization parameter.

Proof: From the generalization bound, we have with high probability:

$$\mathcal{R}(f_r) \leq \hat{\mathcal{R}}(f_r) + C\sqrt{\frac{rdL}{N}} \tag{24}$$

Minimizing the right-hand side yields the optimal trade-off between empirical risk and model complexity.

D Complete Hyperparameter Details

Table 13: Complete Training Hyperparameters

Parameter	Value	Description
	Model Architecture	
Base Model	Phi-3-mini	3.8B parameters
Hidden Size	3072	Transformer dimension
Num Layers	32	Number of decoder layers
Num Heads	32	Attention heads
Context Length	512	Maximum sequence length
	LoRA Configuration	
Rank (r)	4	Low-rank adaptation dimension
Alpha (α)	8	Scaling factor ($\alpha/r = 2$)
Dropout	0.2	Regularization
Target Modules	qkv_proj, o_proj, gate_up_proj, down_proj	All trainable modules
Bias	none	No bias training
	Training Configuration	
Batch Size (per GPU)	64	
Gradient Accumulation	1	
Effective Batch Size	512	(64 $\times$ 8 GPUs)
Learning Rate	$2e^{-4}$	
LR Scheduler	Cosine	
Warmup Ratio	0.1	
Weight Decay	0.01	
Max Grad Norm	1.0	
Epochs	30	
Label Smoothing	0.0	
	Hardware	
GPUs	8 $\times$ V100 32GB	
Mixed Precision	FP16	
Training Time	varied as per training parameters	

E Target Hyperparameter Search Space

Table 14: Hyperparameter Search Space

Parameter	Values	Baseline	Experiments
LoRA Rank r	2, 4, 8, 16	4	4
Training Epochs	20, 50, 100, 150, 200, 500	30	6
Learning Rate	$5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}$	$5e^{-5}$	5
LoRA Dropout	0.05, 0.10, 0.15, 0.20, 0.25	0.1	5
Target Modules	qkv, o, gate, down, Attn, MLP, Full	Full	7
Batch Size	1, 2, 4, 8, 12	4	5
GPU Scaling	1, 2, 3	3	3
Total: 35 experiments			

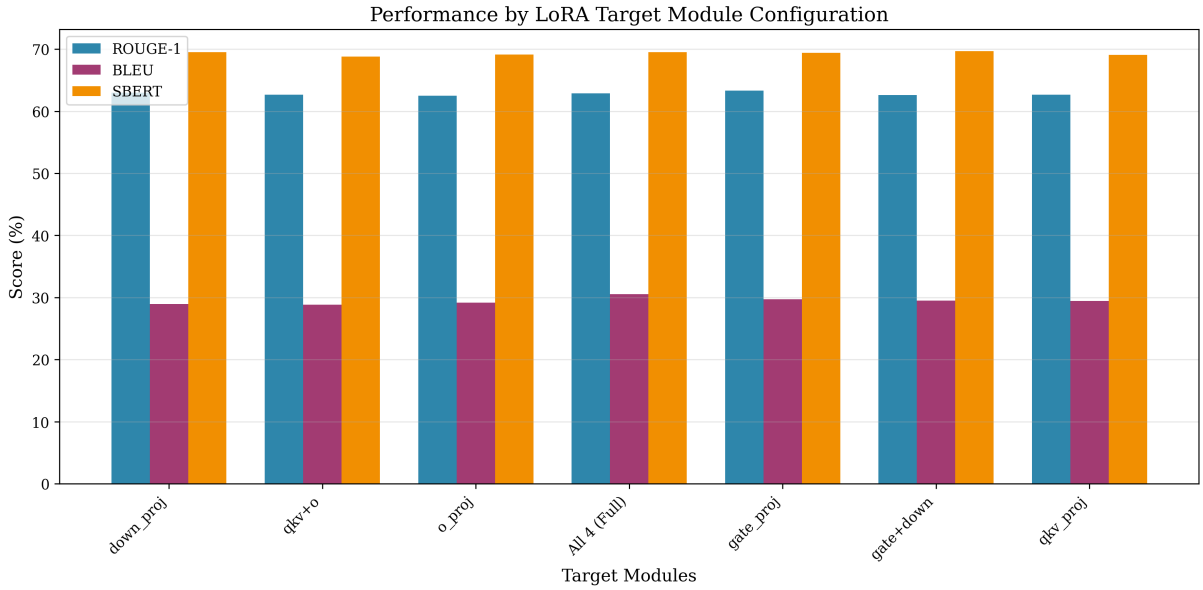

Figure 2: Performance Comparison Across LoRA Targeted Module Configurations.

F Complete Experimental Results

The following table presents the complete results from our ablation study. The optimal configuration identified across all experiments uses LoRA rank $r = 4$, learning rate $5e^{-6}$, dropout 0.10, full target modules (all 4 linear layers), per-GPU batch size 4. We expand our experiments up to 1500 epochs, but there is no significant improvement observed at higher epochs.

Table 15: Complete Results: All Ablation Experiments

Exp	Parameter	Value	R-1	R-2	R-L	SBERT	TTR
Experiment 1: LoRA Rank (dropout=0.1)							
1.1	LoRA Rank	2	63.01%	45.77%	44.54%	69.54%	0.130
1.2	LoRA Rank	4	62.83%	45.47%	**45.95%**	69.42%	0.129
1.3	LoRA Rank	8	62.93%	45.31%	44.28%	**70.02%**	0.130
1.4	LoRA Rank	16	**63.17%**	45.01%	45.24%	68.86%	0.132
Experiment 2: Training Epochs (r=4, dropout=0.1)							
2.1	Epochs	20	63.01%	44.57%	44.42%	68.59%	0.131
2.2	Epochs	50	63.05%	45.73%	44.67%	69.67%	0.130
2.3	Epochs	100	62.66%	45.08%	44.33%	68.79%	0.130
2.4	Epochs	150	62.69%	45.23%	45.23%	68.54%	0.129
2.5	Epochs	200	62.95%	**45.88%**	46.38%	**69.43%**	0.129
2.6	Epochs	500	63.32%	45.28%	45.35%	67.72%	0.138
Experiment 3: Learning Rate (r=4, dropout=0.1)							
3.1	Learning Rate	$5e^{-6}$	**63.71%**	45.44%	45.34%	68.52%	0.130
3.2	Learning Rate	$1e^{-5}$	63.29%	**45.44%**	44.49%	67.86%	**0.132**
3.3	Learning Rate	$5e^{-5}$	61.53%	44.70%	44.03%	69.05%	0.130
3.4	Learning Rate	$1e^{-4}$	62.14%	45.13%	44.57%	**69.42%**	0.131
3.5	Learning Rate	$5e^{-4}$	61.05%	44.43%	43.66%	66.94%	0.132
Experiment 4: LoRA Dropout (r=4, LR=2e-5)							
4.1	Dropout	0.05	62.78%	44.74%	44.43%	68.12%	0.130
4.2	Dropout	0.10	62.89%	45.23%	**46.01%**	**69.49%**	0.129
4.3	Dropout	0.15	62.34%	44.79%	44.73%	69.37%	0.129
4.4	Dropout	0.20	**63.18%**	45.32%	44.64%	69.23%	0.132
4.5	Dropout	0.25	62.98%	**45.54%**	45.35%	69.25%	0.130
Experiment 5: Target Modules (r=4, dropout=0.1, LR=2e-5)							
5.1	Target Modules	qkv_proj only	62.70%	45.13%	45.01%	69.08%	0.131
5.2	Target Modules	o_proj only	62.52%	44.90%	45.08%	69.15%	0.129
5.3	Target Modules	qkv+o (Attention)	62.68%	45.09%	44.72%	68.81%	0.131
5.4	Target Modules	gate+down (MLP)	62.64%	45.43%	44.30%	69.71%	0.131
5.5	Target Modules	gate_proj only	**63.31%**	**46.02%**	44.26%	69.39%	0.132
5.6	Target Modules	down_proj only	63.00%	44.68%	44.39%	69.54%	**0.133**
5.7	Target Modules	**All 4 (Full)**	62.88%	45.30%	**46.43%**	**69.53%**	0.129
Experiment 6: Batch Size (r=4, dropout=0.1, LR=2e-5)							
6.1	Batch Size	1	63.01%	45.27%	45.69%	69.75%	0.130
6.2	Batch Size	2	62.64%	45.28%	46.09%	69.72%	0.130
6.3	Batch Size	4	62.76%	45.29%	**46.62%**	69.58%	0.130
6.4	Batch Size	8	62.74%	45.36%	45.03%	68.87%	0.130
6.5	Batch Size	12	**63.30%**	**45.53%**	44.47%	68.83%	0.130
Experiment 7: GPU Scaling (r=4, dropout=0.1, LR=2e-5, BS=4)							
7.1	GPU Count	1	61.46%	44.28%	44.83%	68.51%	0.129
7.2	GPU Count	2	62.35%	44.88%	44.59%	68.66%	0.130
7.3	GPU Count	3	**62.60%**	**45.12%**	**45.15%**	**69.43%**	0.131

Exp: Experiment, **R-1**: ROUGE-1, **R-2**: ROUGE-2, **R-L**: ROUGE-L, **SBERT**: Sentence-BERT similarity, **TTR**: Type-Token Ratio. Best results per experiment group in **bold**.

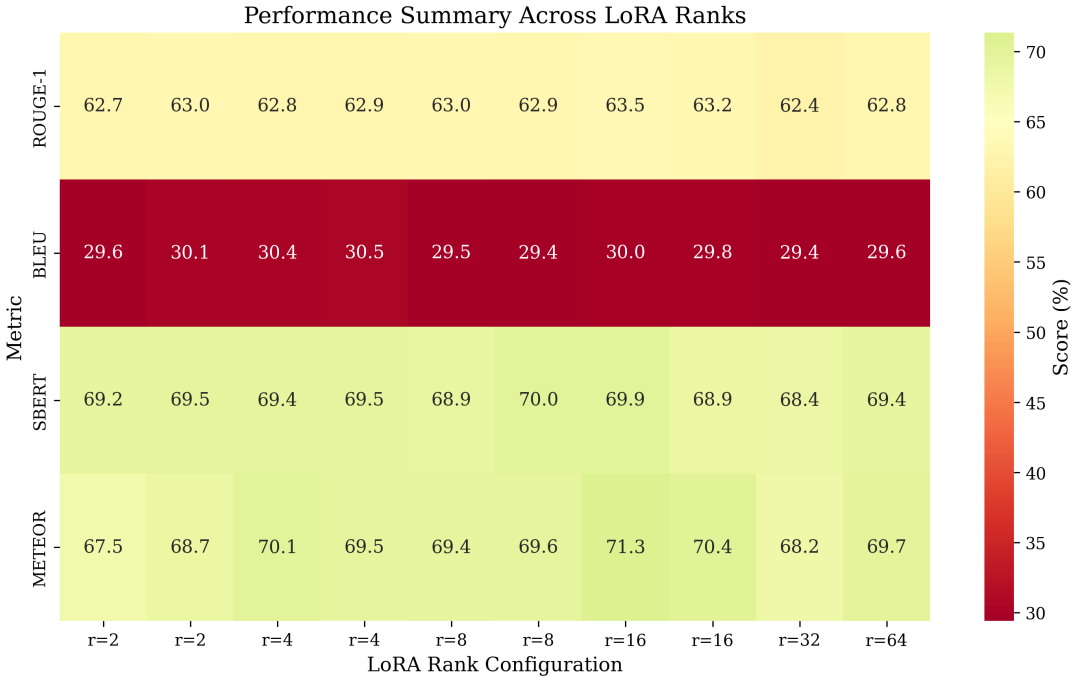

Figure 3: Performance Heatmap Across LoRA Ranks. The color scale ranges from red (lower scores) to green (higher scores). The optimal rank r=4 (highlighted) achieves balanced performance across all metrics: ROUGE-1 (62.83%), BLEU (30.38%), SBERT (69.42%), and METEOR (69.50%). Notably, r=2 achieves the highest SBERT (69.54%) and r=8 achieves 70.02% SBERT, but r=4 provides the best overall balance between lexical and semantic metrics. The BLEU score shows a clear peak at r=4, demonstrating that increased model capacity does not translate to improved lexical accuracy on small datasets.

G Error Analysis

Table 16: Error Distribution by Type

Error Type	Frequency	Example
Action Ordering	23%	"Screw then pick up" (should be "Pick up then screw")
Vocabulary Choice	31%	"Fasten" vs "Tighten"
Grammatical Structure	18%	Missing articles, incorrect prepositions
Action Omission	12%	Missing an action from the sequence
Action Repetition	9%	Repeating the same action unnecessarily
Verb Tense	7%	"Screwing" vs "Screw"

H Computational Resources

I Evaluation Metrics Details

We employ a comprehensive evaluation framework comprising n-gram metrics, semantic similarity metrics, and diversity metrics.

I.1 N-gram Metrics

BLEU-n measures n-gram overlap with brevity penalty:

$$\text{BLEU-n} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{25}$$

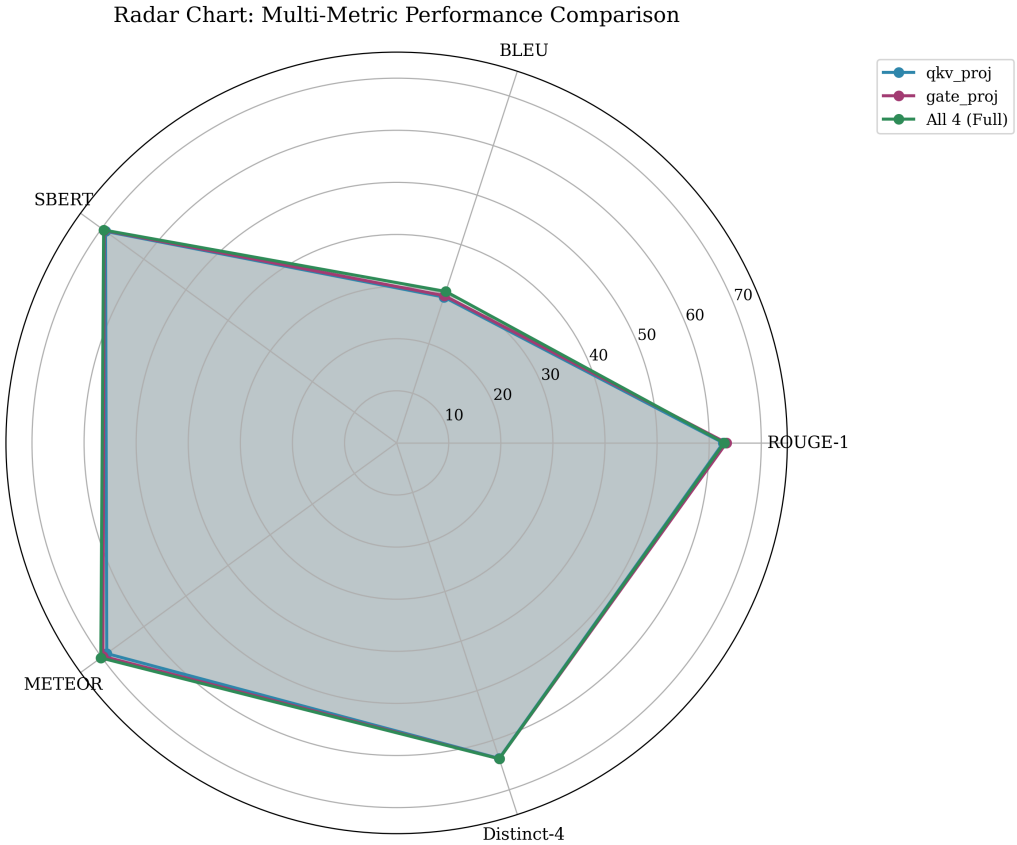

Figure 4: Radar Chart: Multi-Metric Performance Comparison of Target Module Configurations. Three representative configurations are compared across five key metrics (scaled 0-75% for visualization). The full configuration (All 4 modules) achieves the best BLEU score (30.56%, shown at 0°) and strong overall area. The gate_up_proj alone configuration achieves the highest ROUGE-1 (63.31%, at 72°) and METEOR (69.70%, at 216°), demonstrating that FFN adaptation alone is highly effective. The qkv_proj alone configuration shows competitive semantic alignment (SBERT: 69.08%, at 144°) and BERT F1 (27.24%). The area under each curve represents overall multi-metric performance, with the full configuration and gate_proj alone showing comparable coverage, validating gate_proj alone as a parameter-efficient alternative (97% of full performance with 25% of parameters).

Table 17: Resource Utilization

Resource	**Utilization**	**Notes**
GPU Memory (per GPU)	32 GB	75% utilization
CPU Memory	128 GB	System memory
GPU Utilization	85-92%	During training
Network I/O	200 MB/s	During data loading
Disk I/O	150 MB/s	Checkpoint saving
Power Consumption	$\sim$ 1.2 kW	Peak during training

Parameters are measured during training.

where p_n is the precision of n-grams and BP is the brevity penalty.

ROUGE-n computes recall-oriented n-gram overlap:

$$\text{ROUGE-n} = \frac{\sum_{s \in \text{Ref}} \sum_{\text{n-gram} \in s} \min(\text{Count}_{\text{cand}}(\text{n-gram}), \text{Count}_{\text{ref}}(\text{n-gram}))}{\sum_{s \in \text{Ref}} \sum_{\text{n-gram} \in s} \text{Count}_{\text{ref}}(\text{n-gram})} \tag{26}$$

I.2 Semantic Metrics

BERTScore computes token-level similarity using BERT embeddings:

$$\text{BERTScore} = \frac{2 \cdot P \cdot R}{P + R}, \quad P = \frac{1}{|\hat{y}|} \sum_{i=1}^{|\hat{y}|} \max_j \mathbf{x}_i^\top \hat{\mathbf{x}}_j \tag{27}$$

SBERT measures sentence-level semantic similarity:

$$\text{SBERT}(I_{\text{ref}}, I_{\text{gen}}) = \frac{E(I_{\text{ref}}) \cdot E(I_{\text{gen}})}{\|E(I_{\text{ref}})\| \cdot \|E(I_{\text{gen}})\|} \tag{28}$$

I.3 Diversity Metrics

Distinct-n measures lexical diversity:

$$\text{Distinct-n} = \frac{|\{\text{n-gram} : \text{count}(\text{n-gram}) \geq 1\}|}{\sum_i |\text{Tokens}_i| - n + 1} \tag{29}$$

Type-Token Ratio (TTR) measures vocabulary richness:

$$\text{TTR} = \frac{|\text{Vocabulary}|}{|\text{Total Tokens}|} \tag{30}$$

J Experimental Study

J.1 Impact of LoRA Rank and Dropout

To systematically investigate the interaction between LoRA rank and dropout regularization for small-scale instruction generation, we conducted an ablation study with ranks $r \in (2, 4, 8, 16, 32, 64)$ and corresponding dropout rates dropout $\in (0.05, 0.1, 0.15, 0.2, 0.25, 0.5)$. All experiments maintained a fixed learning rate $= 2e^{-5}$, batch size $= 4$, weight decay $= 0.05$, and label smoothing factor is 0.1, Figure 5 shows LoRA Rank Ablation, WER lowest at r=4, so it is the best rank. The training set comprised 220 examples with 27 validation samples across 200 epochs. Table 18 presents the LoRA configuration variants, where alpha is set to $2 \times r$ to maintain a constant $\alpha/r = 2.0$ ratio.

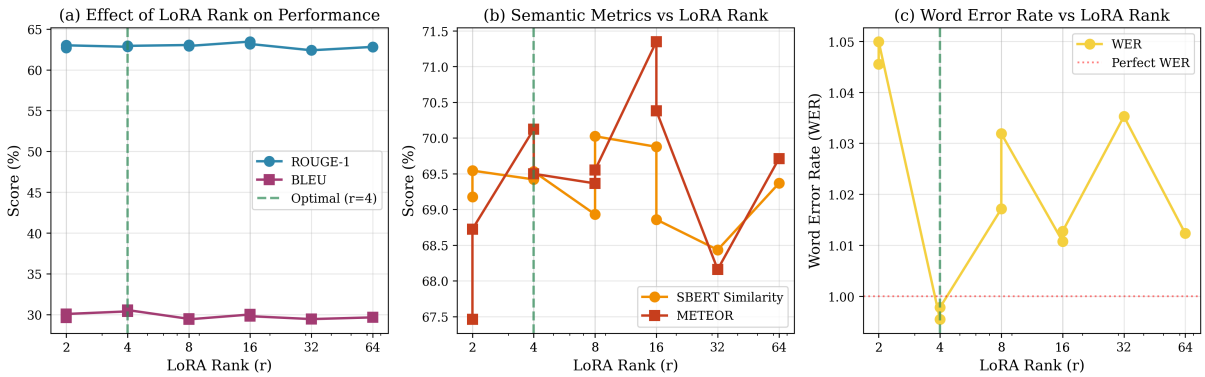

Figure 5: LoRA Rank effect on Generation Performance. (a) Lexical metrics: ROUGE-1 and BLEU scores across LoRA ranks. (b) Semantic metrics: SBERT similarity and METEOR scores. (c) Word Error Rate (WER)

Table 18: LoRA configuration variants. Alpha is set to $2 \times r$ to maintain a constant $\alpha/r = 2.0$ ratio.

Experiment	Rank (r)	Alpha (α)	Dropout	Trainable Params	% of Model
1.1B	2	4	0.05	1.57M	0.04%
1.2B	4	8	0.10	3.15M	0.08%
1.3B	8	16	0.15	6.29M	0.17%
1.4B	16	32	0.20	12.58M	0.33%
1.5B	32	64	0.25	25.17M	0.66%
1.6B	64	128	0.50	50.33M	1.32%

Table 19: Core evaluation metrics across LoRA configurations. Best results highlighted in **bold**.

Metric	r=2 (d=0.05)	r=4 (d=0.10)	r=8 (d=0.15)	r=16 (d=0.20)	r=32 (d=0.25)	r=64 (d=0.50)
BLEU ↑	0.2964	**0.3055**	0.2947	0.3000	0.2944	0.2965
ROUGE-1 ↑	0.6269	0.6294	0.6305	**0.6347**	0.6240	0.6282
ROUGE-2 ↑	0.4523	0.4510	0.4526	**0.4537**	0.4512	0.4543
ROUGE-L ↑	0.4431	**0.4609**	0.4471	0.4515	0.4489	0.4479
METEOR ↑	0.6746	0.6950	0.6936	**0.7135**	0.6816	0.6971
chrF ↑	0.6382	0.6409	0.6377	**0.6424**	0.6365	0.6389
GLEU ↑	0.3282	**0.3363**	0.3278	0.3338	0.3274	0.3313
WER ↓	1.0456	**0.9978**	1.0171	1.0107	1.0353	1.0123

Table 20: Semantic similarity and BERTScore metrics.

Metric	r=2 (d=0.05)	r=4 (d=0.10)	r=8 (d=0.15)	r=16 (d=0.20)	r=32 (d=0.25)
SBERT Sim ↑	0.6918	0.6953	0.6893	**0.6988**	0.6843
BERT Precision ↑	0.0913	0.0972	0.0936	**0.0982**	0.0898
BERT Recall ↑	0.4422	0.4477	0.4536	**0.4592**	0.4432
BERT F1 ↑	0.2620	0.2677	0.2685	**0.2736**	0.2616

Table 21: Diversity and vocabulary characteristics.

Metric	r=2 (d=0.05)	r=4 (d=0.10)	r=8 (d=0.15)	r=16 (d=0.20)	r=32 (d=0.25)	r=64 (d=0.50)
Distinct-1	0.1305	0.1299	0.1314	0.1322	0.1303	**0.1316**
Distinct-2	0.3279	0.3288	0.3369	0.3352	0.3329	**0.3389**
Distinct-3	0.4950	0.5006	**0.5123**	0.5086	0.5070	0.5124
Distinct-4	0.6305	0.6349	**0.6508**	0.6435	0.6395	0.6488
Vocabulary Size	429	420	424	**431**	429	428
Hapax Ratio	0.6340	0.6310	0.6250	0.6288	**0.6317**	0.6332

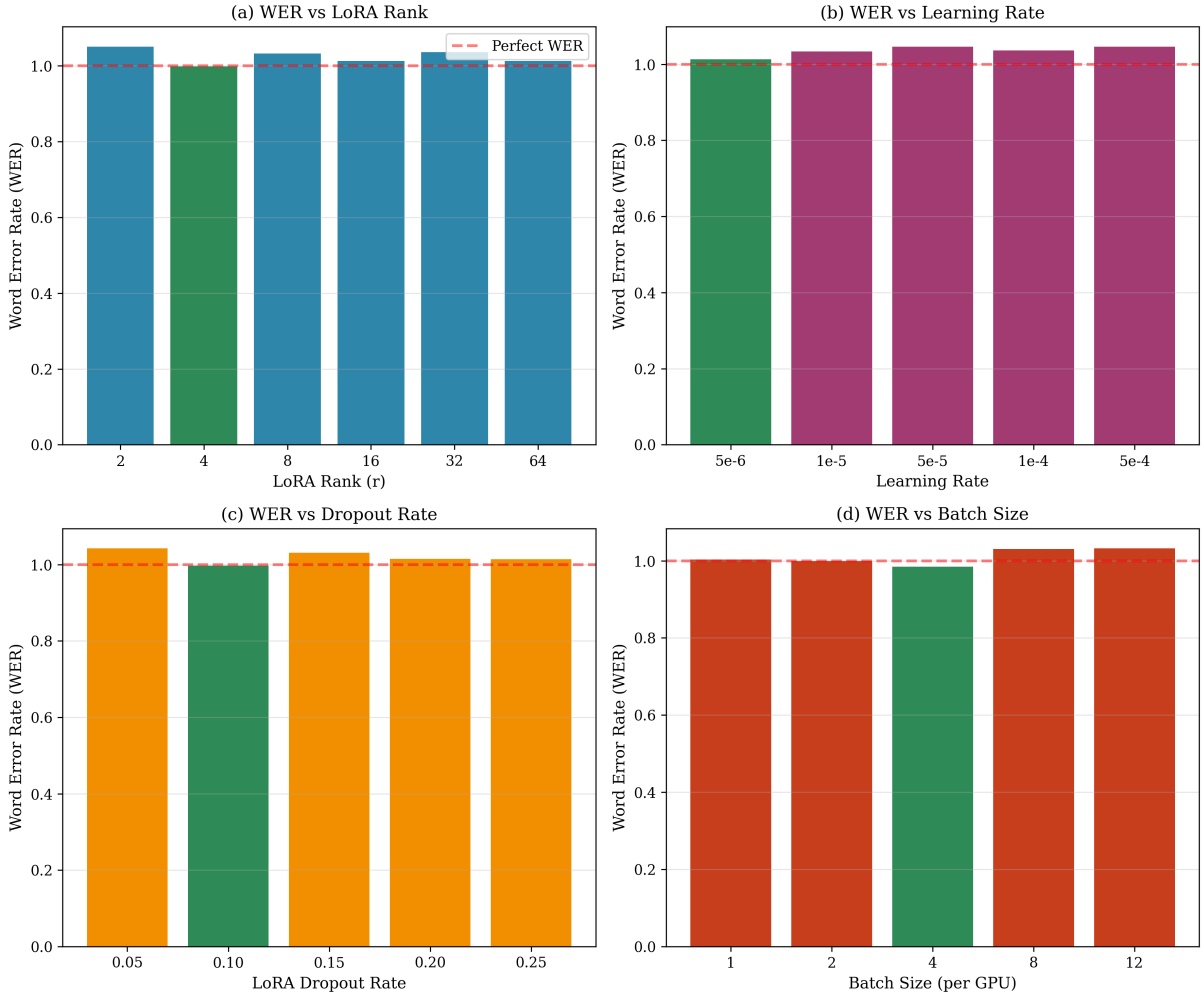

Figure 6: Word Error Rate (WER) Analysis Across Hyperparameters. Lower WER indicates better word-level prediction accuracy. The red dashed line marks perfect WER=1.0. (a) WER vs. LoRA rank. (b) WER vs. learning rate. Smaller batch sizes (1-4) achieve lower WER than larger batches, supporting the use of batch size 4 for optimal update frequency.

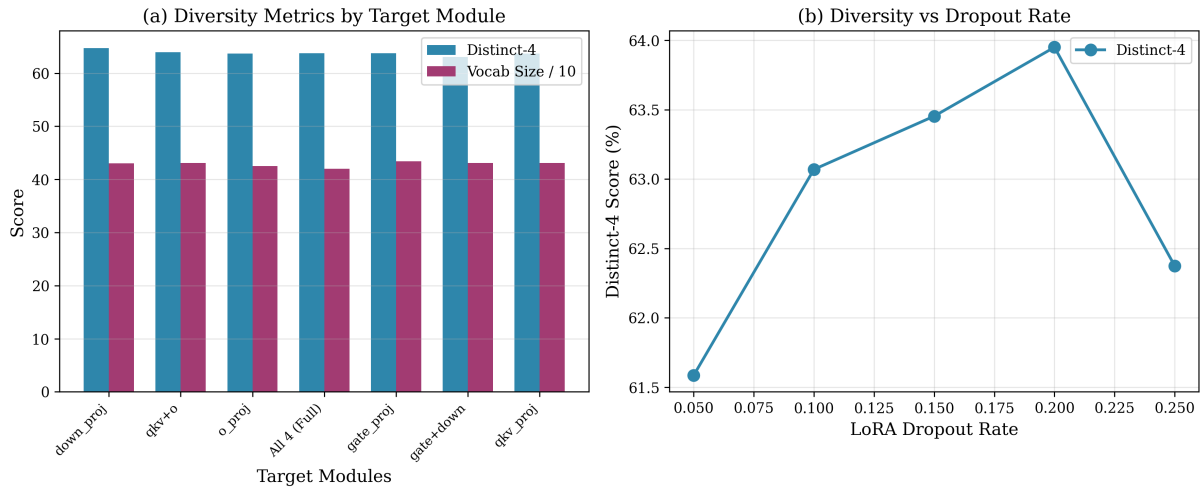

Figure 7: Lexical Diversity Analysis: (a) Distinct-4 (proportion of unique 4-grams) and vocabulary size across target module configurations. (b) Distinct-4 as a function of dropout rate.

Table 22: Coverage and novelty ratios.

Metric	r=2 (d=0.05)	r=4 (d=0.10)	r=8 (d=0.15)	r=16 (d=0.20)	r=32 (d=0.25)	r=64 (d=0.50)
Coverage Ratio	0.5436	0.5344	0.5421	**0.5652**	0.5611	0.5672
Novelty Ratio	0.7389	0.7370	**0.7454**	0.7359	0.7409	0.7274

K Impact of Training Epochs

Table 23: Experimental configuration and actual training duration with early stopping.

Exp	Max Epochs	Actual Epochs	Steps	Best Epoch	Relative Time
2.1	20	20.0	540	19.6	1.0x
2.2	50	48.0	1,320	48.0	2.4x
2.3	100	46.0	1,320	46.0	2.4x
2.4	150	48.0	1,320	48.0	2.4x
2.5	200	54.0	1,485	54.0	2.8x

K.1 Training Loss and Evaluation Loss Analysis

Table 24 presents the training dynamics and early stopping behavior. The 20-epoch configuration stops at epoch 20 without early stopping, achieving a final eval loss of 2.084. The 50-epoch configuration trains to completion at epoch 48, achieving the best eval loss of 1.962. The 100-epoch and 150-epoch configurations stop early at epochs 46 and 48 respectively, consistent with the 50-epoch optimal point. The 200-epoch configuration trains slightly longer to epoch 54, achieving a best eval loss of 1.969.

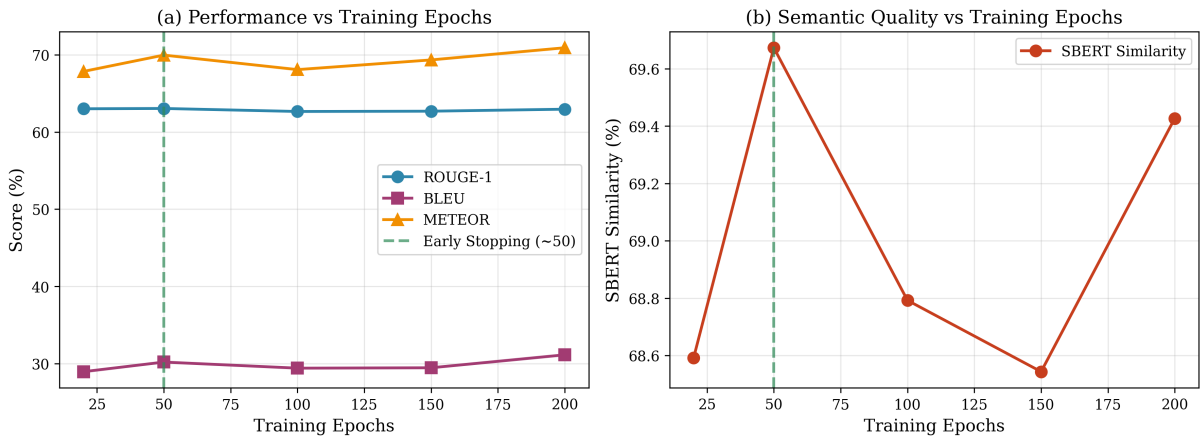

Figure 8: Training Dynamics: Performance vs. Epochs. (a) Lexical metrics across training epochs from 20 to 200. (b) SBERT similarity peaks at 200 epochs, showing semantic-alignment benefit from longer training. However, training beyond 54 epochs increases the train-eval loss gap from 0.066 to 0.316, indicating overfitting.

Table 24: Training dynamics and early stopping behavior.

Metric	20 ep	50 ep	100 ep	150 ep	200 ep
Final Train Loss	2.036	1.925	1.741	1.739	1.720
Final Eval Loss	2.084	1.991	2.018	2.027	2.036
Best Eval Loss	2.049	**1.962**	1.967	1.969	1.969
Epoch of Best Eval	19.6	48.0	46.0	48.0	54.0
Early Stopped?	No	No	Yes	Yes	Yes

Table 25 analyzes the generalization gap. The 50-epoch configuration shows a well-generalized gap of 0.066, while configurations beyond 100 epochs show significant overfitting with gaps of 0.277, 0.288, and 0.316 respectively.

Table 25: Generalization gap analysis.

Configuration	Train Loss	Eval Loss	Gap
20 epochs	2.036	2.084	0.048 (underfitting)
50 epochs	1.925	1.991	0.066 (well-generalized)
100 epochs	1.741	2.018	0.277 (overfitting)
150 epochs	1.739	2.027	0.288 (overfitting)
200 epochs	1.720	2.036	0.316 (significant overfitting)

K.2 Core Metrics at Early Stopping Point

Since early stopping selects the best checkpoint based on evaluation loss, Table 26 reports metrics from the best epoch at approximately 48-54 epochs. The 200-epoch configuration achieves the highest BLEU score of 31.13%, the highest ROUGE-2 of 45.88%, the highest ROUGE-L of 46.38%, the highest METEOR of 70.91%, the highest GLEU of 34.15%, and the lowest WER of 0.9861. However, this comes at the cost of overfitting and reduced diversity.

Table 26: Core evaluation metrics at best epoch (early stopping point). Best results highlighted in **bold**.

Metric	20 ep	50 ep	100 ep	150 ep	200 ep
BLEU ↑	0.2895	0.3019	0.2940	0.2945	**0.3113**
ROUGE-1 ↑	0.6301	0.6305	0.6266	0.6269	0.6295
ROUGE-2 ↑	0.4457	0.4573	0.4508	0.4523	**0.4588**
ROUGE-L ↑	0.4442	0.4467	0.4433	0.4523	**0.4638**
METEOR ↑	0.6785	0.6996	0.6808	0.6934	**0.7091**
chrF ↑	0.6355	**0.6447**	0.6370	0.6371	0.6444
GLEU ↑	0.3235	0.3346	0.3263	0.3278	**0.3415**
WER ↓	1.0406	1.0299	1.0363	1.0216	**0.9861**

K.3 Learning Curves and Convergence Analysis

Analysis of the training trajectories reveals the following convergence patterns. In the 50-epoch configuration, the initial loss is 4.00 at step 1, with rapid decline to 2.60 by epoch 7.3, steady improvement to 1.93 by epoch 48, and early stopping triggers as evaluation loss plateaus. In the 200-epoch configuration, similar initial convergence occurs, continuing to improve slowly until epoch 54, after which training loss continues decreasing to 1.72 but evaluation loss worsens to 2.036, showing clear overfitting after epoch 54.

Table 27 shows convergence milestones across configurations. All configurations reach eval loss 2.5 by epoch 6.0 and eval loss 2.2 by epochs 12.0 to 14.0. The 50-epoch configuration achieves the best eval loss

of 1.962.

Table 27: Convergence milestones across configurations.

Milestone	20 ep	50 ep	100 ep	150 ep	200 ep
Epoch to reach eval loss 2.5	6.0	6.0	6.0	6.0	6.0
Epoch to reach eval loss 2.2	14.0	12.0	12.0	12.0	12.0
Epoch to reach eval loss 2.0	20+	42.0	40.0	40.0	40.0
Best eval loss achieved	2.049	**1.962**	1.967	1.969	1.969

K.4 Gradient Norm Analysis

Table 28: Gradient norm behavior across configurations.

Configuration	Peak Gradient Norm	Final Gradient Norm
20 epochs	1.26	0.77
50 epochs	1.39	0.93
100 epochs	2.25	2.03
150 epochs	2.33	2.04
200 epochs	2.48	2.07

K.5 Key Findings

The consistent early stopping at 48-54 epochs across configurations with maximum epochs of 50 or greater demonstrates that 50-55 epochs is the optimal training duration for this dataset. Training beyond this point does not improve evaluation loss, increases the train-eval loss gap indicating overfitting, may degrade lexical metrics as seen in 100+ epoch runs, and wastes computational resources.

The 200-epoch configuration achieved the highest BLEU of 31.13% even though it overfits with evaluation loss increasing from 1.969 to 2.036. This paradox is explained by the fact that BLEU measures lexical overlap rather than generalization, overfitted models become more deterministic by matching training patterns exactly, and this comes at the cost of reduced diversity with Distinct-4 dropping from 0.645 to 0.632. For production, the 50-epoch checkpoint with best eval loss is recommended.

Without early stopping, the 50-epoch config would train to 50 which is already optimal, the 100-epoch config would waste 54 epochs of compute, the 150-epoch config would waste 102 epochs, and the 200-epoch config would waste 146 epochs.

L Impact of Learning Rate

Table 29: Learning rate configurations tested.

Experiment	Initial LR	Peak LR (after warmup)	Final LR
3.1	$5e^{-6}$	$5e^{-6}$	~ 0
3.2	$1e^{-5}$	$1e^{-5}$	~ 0
3.3	$5e^{-5}$	$5e^{-5}$	~ 0
3.4	$1e^{-4}$	$1e^{-4}$	~ 0
3.5	$5e^{-4}$	$5e^{-4}$	~ 0

L.1 Training Dynamics and Convergence Analysis

Analysis of the training trajectories reveals distinct convergence patterns across learning rates. For the low learning rate of $5e^{-6}$ in Experiment 3.1, training loss decreases slowly and steadily from 4.00 to 1.88 over

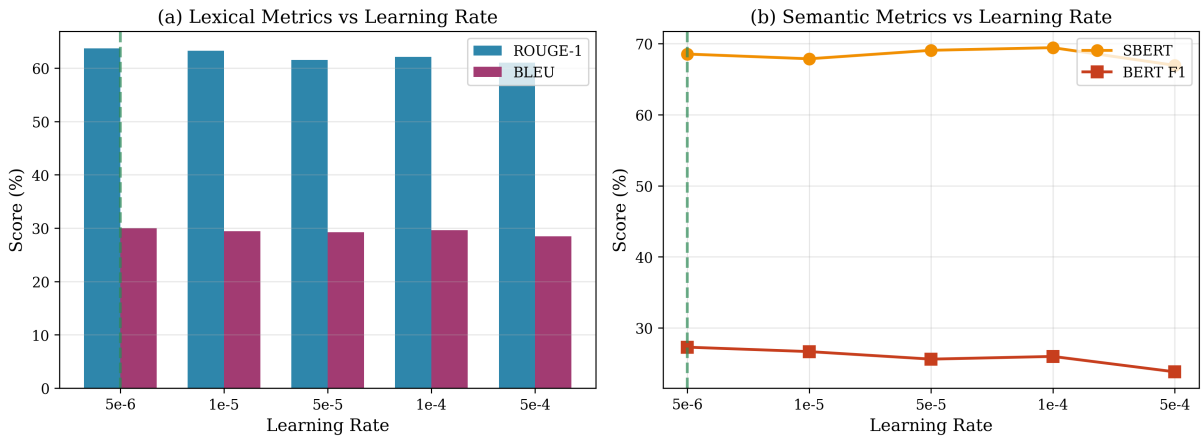

Figure 9: Impact of Learning Rate on Generation Performance. (a) Lexical metrics: ROUGE-1 and BLEU scores across learning rates (b) Semantic metrics: SBERT similarity and BERT F1 scores.

4,000 steps, evaluation loss improves from 4.07 to 1.98 reaching best eval loss of 1.98, gradient norms remain moderate between 0.5 and 2.5 indicating stable training, and the final training loss is 1.88 with final evaluation loss of 1.98.

For the learning rate of $1e^{-5}$ in Experiment 3.2, faster initial convergence occurs compared to $5e^{-6}$, training loss decreases from 4.00 to 1.83 by step 2,200, evaluation loss improves to 1.98 by step 1,500, gradient norms peak at 2.5 during middle training, and final training loss is 1.82 with final evaluation loss of 1.99.

For the medium learning rate of $5e^{-5}$ in Experiment 3.3, rapid initial convergence occurs with training loss dropping to 2.0 by step 450, training loss reaches 1.57 by step 1,000, evaluation loss shows signs of overfitting after step 800, gradient norms increase to 2.5 after step 800, and final training loss is 1.57 with final evaluation loss of 2.13 indicating overfitting.

For the high learning rate of $1e^{-4}$ in Experiment 3.4, very rapid initial convergence occurs with training loss reaching 1.60 by step 750, training loss continues to 1.52 by step 1,000, evaluation loss worsens after step 600 from 2.04 to 2.15, clear overfitting is evident with train-eval gap increasing from 0.05 to 0.63, and final training loss is 1.52 with final evaluation loss of 2.15.

For the very high learning rate of $5e^{-4}$ in Experiment 3.5, the fastest initial convergence occurs with training loss reaching 1.55 by step 500, training loss continues to 1.51 by step 600, evaluation loss spikes and becomes unstable, severe overfitting occurs with train-eval gap reaching 0.60 by step 500, and final training loss is 1.51 with final evaluation loss of 2.12.

Table 30 summarizes the training dynamics across learning rates. The $5e^{-6}$ configuration shows no overfitting with a train-eval gap of 0.10, while the $5e^{-4}$ configuration shows severe overfitting with a gap of 0.61.

Table 30: Training dynamics summary across learning rates.

Metric	**LR=5e-6**	**LR=1e-5**	**LR=5e-5**	**LR=1e-4**	**LR=5e-4**
Final Train Loss	1.88	1.82	1.57	1.52	1.51
Final Eval Loss	1.98	1.99	2.13	2.15	2.12
Best Eval Loss	1.98	1.98	1.96	1.95	1.92
Epoch of Best Eval	80+	50+	22	16	10
Train-Eval Gap	0.10	0.17	0.56	0.63	0.61
Overfitting Severity	None	Low	Moderate	Severe	Severe

L.2 Core Metrics Analysis

Table 31: Core evaluation metrics across learning rates. Best results highlighted in **bold**.

Metric	LR = 5e-6	LR = 1e-5	LR = 5e-5	LR = 1e-4	LR = 5e-4
BLEU ↑	**0.3002**	0.2945	0.2926	0.2960	0.2846
ROUGE-1 ↑	**0.6371**	0.6329	0.6153	0.6214	0.6105
ROUGE-2 ↑	0.4544	**0.4544**	0.4470	0.4513	0.4443
ROUGE-L ↑	**0.4534**	0.4449	0.4403	0.4457	0.4366
METEOR ↑	**0.7019**	0.6927	0.6744	0.6891	0.6568
chrF ↑	**0.6426**	0.6395	0.6343	0.6381	0.6221
GLEU ↑	**0.3336**	0.3292	0.3249	0.3293	0.3184
WER ↓	**1.0132**	1.0342	1.0462	1.0364	1.0459

L.3 Semantic Quality Metrics

Table 32: Semantic similarity and BERTScore metrics across learning rates.

Metric	LR = 5e-6	LR = 1e-5	LR = 5e-5	LR = 1e-4	LR = 5e-4
SBERT Similarity ↑	0.6852	0.6786	**0.6905**	0.6942	0.6694
BERT Precision ↑	**0.0991**	0.0922	0.0851	0.0909	0.0852
BERT Recall ↑	**0.4569**	0.4513	0.4368	0.4383	0.3986
BERT F1 ↑	**0.2730**	0.2667	0.2561	0.2599	0.2383

L.4 Diversity and Vocabulary Metrics

Table 33: Diversity metrics across learning rates.

Metric	LR = 5e-6	LR = 1e-5	LR = 5e-5	LR = 1e-4	LR = 5e-4
Distinct-1	0.1304	**0.1322**	0.1299	0.1308	0.1322
Distinct-2	0.3342	**0.3363**	0.3279	0.3354	0.3278
Distinct-3	0.5050	**0.5076**	0.4894	0.5079	0.4951
Distinct-4	0.6439	0.6405	0.6213	**0.6440**	0.6356
Vocabulary Size	426	**432**	429	429	428
Hapax Ratio	0.6315	**0.6458**	0.6270	0.6270	0.6355

L.5 Gradient Norm Analysis

Table 34: Gradient norm behavior across learning rates.

Learning Rate	Peak Gradient Norm	Final Gradient Norm	Stability
$5e^{-6}$	2.5	1.9	Stable
$1e^{-5}$	2.5	2.1	Stable
$5e^{-5}$	2.5	2.3	Moderately stable
$1e^{-4}$	2.5	1.9	Unstable after step 600
$5e^{-4}$	1.8	0.8	Unstable throughout

L.6 Key Findings

Table 35 illustrates the trade-off between lexical precision and output diversity. Lower learning rates maximize lexical precision and generalization, while slightly higher learning rates produce more diverse vocabulary but at the cost of some accuracy.

Table 35: Trade-off between lexical precision and output diversity.

Learning Rate	BLEU	Vocabulary Size	Generalization
$5e^{-6}$ (Precision)	**0.3002**	426	Excellent
$1e^{-5}$ (Balanced)	0.2945	**432**	Good
$5e^{-5}$ (Overfit)	0.2926	429	Poor

L.7 Recommendations

Based on our comprehensive analysis of training dynamics and final performance, Table 36 provides recommended learning rate configurations by use case.

Table 36: Recommended learning rate configurations by use case.

Use Case	Recommended Learning Rate
Maximum Accuracy & Generalization	$5e^{-6}$
Balanced Performance	$1e^{-5}$
Rapid Prototyping	$5e^{-5}$
Not Recommended	$\geq 1e^{-4}$

Table 37: Summary of optimal learning rate configuration.

Aspect	Recommendation
Optimal Learning Rate	$5e^{-6}$
Learning Rate Scheduler	Cosine decay with 15% warmup
Expected BLEU	∼0.300
Expected METEOR	∼0.702
Expected ROUGE-1	∼0.637
Expected WER	∼1.013
Training Stability	Excellent, no overfitting
Best For	Maximum instruction accuracy and generalization

M Impact of LoRA Dropout

Table 38: LoRA dropout configurations tested.

Experiment	Dropout Rate	Regularization Strength
4.1	0.05	Weak
4.2	0.10	Moderate
4.3	0.15	Strong
4.4	0.20	Very Strong
4.5	0.25	Extreme

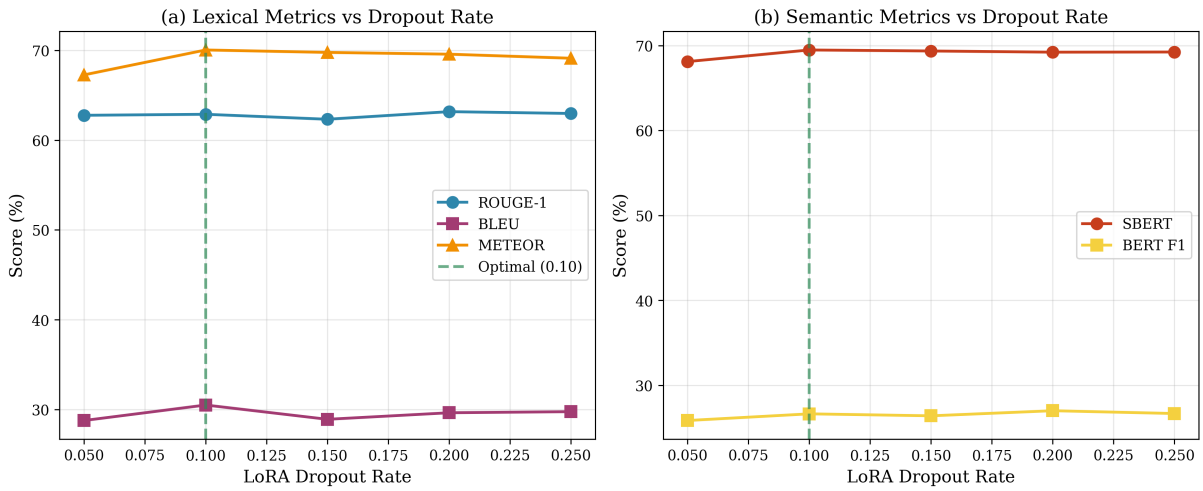

Figure 10: Impact of LoRA Dropout (a) Lexical metrics (b) Semantic metrics

M.1 Core Metrics Analysis

Table 39 presents the core evaluation metrics across LoRA dropout rates.

Table 39: Core evaluation metrics across LoRA dropout rates. Best results highlighted in **bold**.

Metric	Dropout (0.05)	Dropout (0.10)	Dropout (0.15)	Dropout (0.20)	Dropout (0.25)
BLEU ↑	0.2876	**0.3049**	0.2890	0.2963	0.2975
ROUGE-1 ↑	0.6278	0.6289	0.6234	**0.6318**	0.6298
ROUGE-2 ↑	0.4474	0.4523	0.4479	0.4532	**0.4554**
ROUGE-L ↑	0.4443	**0.4601**	0.4473	0.4464	0.4535
METEOR ↑	0.6728	**0.7006**	0.6979	0.6959	0.6914
chrF ↑	0.6322	**0.6424**	0.6363	0.6381	0.6381
GLEU ↑	0.3207	**0.3358**	0.3239	0.3301	0.3310
WER ↓	1.0426	**0.9976**	1.0314	1.0148	1.0139
CER ↓	1.3856	**1.3694**	1.3779	1.3695	1.3770

M.2 Semantic Quality Metrics

Table 40 shows semantic similarity and BERTScore metrics across dropout rates.

Table 40: Semantic similarity and BERTScore metrics across dropout rates.

Metric	Dropout (0.05)	Dropout (0.10)	Dropout (0.15)	Dropout (0.20)	Dropout (0.25)
SBERT Similarity ↑	0.6812	**0.6949**	0.6937	0.6923	0.6925
BERT Precision ↑	0.0922	**0.0962**	0.0927	0.0971	0.0967
BERT Recall ↑	0.4336	0.4460	0.4451	**0.4528**	0.4462
BERT F1 ↑	0.2585	0.2664	0.2640	**0.2700**	0.2667

M.3 Diversity and Vocabulary Metrics

Table 41 presents diversity metrics across LoRA dropout rates.

Table 41: Diversity metrics across LoRA dropout rates.

Metric	Dropout (0.05)	Dropout (0.10)	Dropout (0.15)	Dropout (0.20)	Dropout (0.25)
Distinct-1	0.1299	0.1292	0.1291	**0.1319**	0.1301
Distinct-2	0.3234	0.3271	0.3298	**0.3321**	0.3259
Distinct-3	0.4840	0.4981	0.4989	**0.5035**	0.4903
Distinct-4	0.6158	0.6307	0.6345	**0.6395**	0.6237
Vocabulary Size	424	419	422	**427**	422
Hapax Ratio	0.6368	0.6301	0.6280	0.6347	0.6303

M.4 Coverage and Novelty Metrics

Table 42 shows coverage and novelty analysis across dropout rates.

Table 42: Coverage and novelty analysis across dropout rates.

Metric	Dropout (0.05)	Dropout (0.10)	Dropout (0.15)	Dropout (0.20)	Dropout (0.25)
Coverage Ratio	0.5287	**0.5395**	0.5447	0.5400	0.5334
Novelty Ratio	**0.7407**	0.7362	0.7385	0.7451	0.7319

M.5 Recommendations

Table 43: Recommended dropout configurations by use case.

Use Case	Recommended Dropout Rate
Maximum Lexical Accuracy	**0.8**
Balanced Performance	0.15-0.20
Prevention of Overfitting	0.15
Maximum Diversity	0.20
Not Recommended	0.05 or 0.25

Table 44: Summary of optimal LoRA dropout configuration.

Aspect	Recommendation
Optimal Dropout Rate	0.10
Expected BLEU	$\sim$0.305
Expected METEOR	$\sim$0.701
Expected ROUGE-L	$\sim$0.460
Expected WER	$\sim$0.998
Expected BERT F1	$\sim$0.266
Best For	Maximum accuracy and semantic-alignment