

Efficient Fine-Tuning of GPT-2: LoRA, Hyperparameter Search, and Scaling for Paraphrase Detection and Sonnet Generation

Stanford CS224N Default Project

Biswas Shrestha

Department of Computer Science

Stanford University

biswassh@stanford.edu

Abstract

We implement GPT-2 [1] from scratch and investigate efficient fine-tuning strategies across three downstream tasks: sentiment classification, paraphrase detection, and sonnet generation. Our main extension is Low-Rank Adaptation (LoRA) [2], which we implement from scratch and evaluate against full fine-tuning on paraphrase detection (Quora, 141K examples) and sonnet generation (143 training Shakespeare sonnets). For paraphrase detection, full fine-tuning achieves 0.895 dev accuracy (exceeding the staff baseline of 0.882), while LoRA with rank $r = 4$ reaches 0.839, revealing a 5.6-point gap that shrinks as rank increases but plateaus at $r = 8$. A learning rate sweep identifies 2×10^{-5} as the sweet spot for full fine-tuning. For sonnet generation, all LoRA configurations and a GPT-2 Medium scaling experiment fail to match the full fine-tuning baseline (chrF 28.077 dev / 41.199 test), which essentially ties the staff baseline. We trace this failure to the extreme data scarcity (143 training sonnets), where LoRA's rank constraint limits the model's ability to adapt its generation behavior. Our results highlight a key finding: LoRA's efficiency advantage holds on large datasets but breaks down on very small ones, where full fine-tuning's additional capacity matters.

1 Key Information

- Mentor: Ali Sartaz Khan
- External Collaborators: None
- Sharing project: No

2 Introduction

Fine-tuning large pretrained language models is now the standard approach for NLP tasks, but it comes with a real cost: updating all parameters of a 124M-parameter model like GPT-2 is slow, memory-intensive, and prone to overfitting on small datasets. Parameter-efficient fine-tuning (PEFT) methods like LoRA [2] address this by freezing the pretrained weights and learning only a small number of additional parameters, typically less than 1% of the original model.

This project builds GPT-2 from scratch and explores how full fine-tuning compares to LoRA across two very different data regimes: paraphrase detection on the large Quora dataset (141K examples) and sonnet generation on a tiny Shakespeare corpus (143 training sonnets). These two settings let us test a natural hypothesis: LoRA should work well when data is plentiful (since the pretrained weights already capture most of what is needed), but may struggle when data is scarce and the model needs to substantially shift its behavior.

We also investigate hyperparameter sensitivity for full fine-tuning, running a learning rate sweep on paraphrase detection, and attempt to improve sonnet generation by scaling to GPT-2 Medium with LoRA. Our results confirm the hypothesis about data regime: LoRA closes most of the gap on paraphrase but fails entirely on sonnets, and GPT-2 Medium with LoRA actually performs worse than GPT-2 small with full fine-tuning on the tiny sonnet dataset.

Beyond the accuracy numbers, we find that the chrF metric used for sonnet evaluation has a quirk that makes it reward longer outputs with Shakespearean vocabulary overlap rather than coherent generation, which complicates the interpretation of our sonnet results. We discuss this in detail in the analysis section.

3 Related Work

GPT-2 and decoder-only transformers. GPT-2 [1] is a decoder-only transformer trained on 40GB of web text with a next-token prediction objective. It uses pre-norm residual connections (layer norm before each sub-layer), GELU activations, and weight tying between the input embedding and the output projection. The original Attention is All You Need paper [3] introduced the transformer architecture; GPT-2 adapts it for autoregressive language modeling.

Parameter-efficient fine-tuning. Full fine-tuning updates all model parameters, which is expensive and can overfit on small datasets. Adapter methods [4] insert small trainable modules between transformer layers while freezing the rest. LoRA [2] takes a different approach: it decomposes the weight update ΔW as a product of two low-rank matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d}$, keeping the original weights frozen. This is mathematically equivalent to adapters but adds no inference latency since BA can be merged with W after training. Prefix tuning [5] and prompt tuning [6] are alternative PEFT methods that prepend trainable tokens to the input rather than modifying weights.

Paraphrase detection. The Quora Question Pairs dataset [7] is a standard benchmark for semantic equivalence. Prior work uses cross-encoder architectures with BERT [8] or RoBERTa, which process both sentences jointly. We instead use a cloze-style formulation with GPT-2, treating the task as next-token prediction over “yes” and “no” tokens, which is more natural for a decoder-only model.

Language model fine-tuning for creative text. Fine-tuning language models for stylistic generation (poetry, song lyrics) on small corpora is known to be difficult [9]. The small dataset size makes overfitting a major concern, and automatic metrics like chrF [10] may not capture the quality of creative output well.

4 Approach

4.1 Base GPT-2 Implementation

We implement GPT-2 from scratch following the architecture in [1]. The model has 12 transformer layers, 12 attention heads, and hidden size $d = 768$. Each layer uses pre-norm ordering: layer normalization is applied before both the self-attention and feed-forward sub-layers, and the output is added back via a residual connection.

Causal self-attention. Attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V$$

where M is a causal mask that sets future positions to $-10,000$ before the softmax, and padding positions are also masked additively using the same large negative value. After the softmax, we apply dropout and multiply by V , then reshape back to $[B, T, d]$.

Feed-forward network. Each layer’s FFN uses two linear projections with GELU activation in between: $\text{FFN}(x) = W_2 \cdot \text{GELU}(W_1x + b_1) + b_2$, where $W_1 \in \mathbb{R}^{d \times 4d}$ and $W_2 \in \mathbb{R}^{4d \times d}$.

Weight tying. The output projection $h \mapsto hW_E^\top$ shares weights with the input embedding matrix W_E , following the original GPT-2 design. We load pretrained weights from HuggingFace and verified our implementation against theirs using the provided sanity check (max absolute difference < 0.1).

AdamW optimizer. We implement AdamW from scratch with bias-corrected moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \quad \theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \alpha \lambda \theta_{t-1}$$

where λ is the weight decay coefficient applied directly to the parameters (decoupled from the gradient update).

4.2 Task Formulations

Sentiment classification. We add a linear head over the last non-padding token’s hidden state, producing logits over 5 classes (SST) or 2 classes (CFIMDB), trained with cross-entropy loss. We compare two training regimes: freezing the backbone and training only the head (“last-linear-layer”) vs. updating all parameters (“full-model”).

Paraphrase detection. We use a cloze-style formulation. Given sentence pair (s_1, s_2) , we construct the prompt:

Is " s_1 " a paraphrase of " s_2 "? Answer "yes" or "no":

and use the weight-tied output projection to produce logits over the full vocabulary (50,257 tokens) at the final position. The model is trained with cross-entropy loss to predict the BPE token for “yes” (ID 8505) or “no” (ID 3919). At inference, we take the argmax over all vocabulary tokens; a correct prediction is one where the argmax is the expected yes/no token. This formulation is natural for a decoder-only model and avoids adding any task-specific head.

Sonnet generation. We fine-tune GPT-2 on Shakespeare’s sonnets using standard next-token prediction: the model predicts token $t + 1$ given tokens $1, \dots, t$. At inference, we use top- p (nucleus) sampling with $p = 0.9$ and temperature $\tau = 1.2$, generating up to 256 tokens per sonnet.

4.3 LoRA Implementation

We implement LoRA [2] from scratch as a drop-in replacement for `nn.Linear`. For a pretrained weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the LoRA forward pass is:

$$y = Wx + \frac{\alpha}{r} B A x$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$ is initialized to zero, $A \in \mathbb{R}^{r \times d_{\text{in}}}$ is initialized with Kaiming uniform, and α is a scaling hyperparameter (we set $\alpha = r$ so the effective scale is 1). The pretrained weights W and bias are frozen (`requires_grad=False`); only A and B are trained.

We apply LoRA to five weight matrices per transformer layer: the query (W_Q), key (W_K), value (W_V), attention output (W_O), and FFN output (W_2) projections. This covers all major linear transformations except the FFN up-projection W_1 . With rank r and 12 layers, the total trainable parameters are approximately $12 \times 5 \times 2 \times 768 \times r = 92,160r$. For $r = 4$, this is about 368K parameters, compared to 124M for full fine-tuning (0.30%).

5 Experiments

5.1 Data

We use four datasets. **SST** [11] has 8,544 training and 1,101 dev examples for 5-class sentiment. **CFIMDB** has 1,701 training and 245 dev examples for binary sentiment. **Quora Question Pairs** [7] has 141,506 training and 20,215 dev pairs for paraphrase detection. **Shakespeare Sonnets** has 143 training and 12 dev sonnets for generation; the tiny size of this dataset is a central challenge.

5.2 Evaluation

Sentiment and paraphrase are evaluated by accuracy (fraction of correctly classified examples). Sonnet generation is evaluated by chrF [10], a character n -gram F-score that measures overlap between generated and reference text. We report both dev and test leaderboard scores for paraphrase and sonnet tasks.

5.3 Experimental Details

All experiments run on a single NVIDIA A100 GPU via Google Colab. For paraphrase detection, we use batch size 8 and train for 3–5 epochs. For sentiment, batch size 64 (SST) or 8 (CFIMDB) and 10 epochs. For sonnet generation, batch size 8 and 10 epochs. All models use our from-scratch AdamW implementation with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0.01.

For the LoRA paraphrase rank sweep, we fix $\text{lr} = 10^{-4}$ and sweep $r \in \{2, 4, 8\}$ (we skip $r = 16$ after confirming the plateau at $r = 8$). For the learning rate sweep on full fine-tuning, we compare $\text{lr} \in \{10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$. For sonnet LoRA experiments, we tried ranks $r \in \{4, 16\}$ with learning rates $\{5 \times 10^{-6}, 10^{-5}, 10^{-4}\}$ and multiple matrix subsets. For the GPT-2 Medium experiment, we use the 345M parameter model with 24 layers, 16 heads, and $d = 1024$, with LoRA $r = 16$ on all 6 matrix types and $\text{lr} = 10^{-4}$.

5.4 Results

Sentiment classification. Table 1 shows sentiment results. Full-model fine-tuning outperforms the frozen-backbone approach on both datasets. The CFIMDB full-model result (0.980) exceeds the staff baseline (0.976), while SST (0.497) is slightly below the staff baseline (0.513), likely because SST’s 5-class fine-grained distinctions require more training or a higher learning rate.

Dataset	Method	Dev Acc	Staff Baseline
SST (5-class)	Last-linear-layer	0.454	0.462
SST (5-class)	Full-model	0.497	0.513
CFIMDB (binary)	Last-linear-layer	0.873	0.861
CFIMDB (binary)	Full-model	0.980	0.976

Table 1: Sentiment classification results. Full-model uses $\text{lr} = 10^{-5}$; last-linear-layer uses $\text{lr} = 10^{-3}$.

Paraphrase detection. Table 2 summarizes all paraphrase results. Full fine-tuning with $\text{lr} = 2 \times 10^{-5}$ achieves the best dev accuracy of 0.895 and test accuracy of 0.874, both exceeding the staff baselines (0.882 dev, 0.856 test). The LoRA rank sweep shows a clear trend: accuracy improves from 0.774 at $r = 2$ to 0.839 at $r = 4$, then plateaus at 0.840 for $r = 8$. The gap between the best LoRA result (0.840) and full fine-tuning (0.895) is 5.5 points, which is substantial but not surprising given that LoRA at $r = 4$ uses only 0.30% of the parameters.

Method	Config	Trainable Params	Dev Acc	Test Acc
Full fine-tuning	$\text{lr} = 10^{-5}$, 5 ep	124M	0.894	–
Full fine-tuning	$\text{lr} = 2 \times 10^{-5}$, 3 ep	124M	0.895	0.874
Full fine-tuning	$\text{lr} = 5 \times 10^{-5}$, 3 ep	124M	0.890	–
LoRA $r = 2$	$\text{lr} = 10^{-4}$, 5 ep	184K	0.774	–
LoRA $r = 4$	$\text{lr} = 10^{-4}$, 5 ep	368K	0.839	–
LoRA $r = 8$	$\text{lr} = 10^{-4}$, 5 ep	737K	0.840	–
Staff baseline	–	–	0.882	0.856

Table 2: Paraphrase detection results on Quora Question Pairs. All LoRA models apply low-rank updates to $W_Q, W_K, W_V, W_O, W_{\text{FFN-out}}$ in all 12 layers.

Sonnet generation. Table 3 shows sonnet results. The full fine-tuning baseline (chrF 28.077 dev, 41.199 test) essentially ties the staff baseline (41.208 test). Every LoRA configuration and the GPT-2

Medium experiment scored worse. The best LoRA result (not submitted to test) was from $r = 16$ with all 6 matrix types, but even that produced incoherent output. The GPT-2 Medium + LoRA result (23.948 dev) was the worst of all configurations.

Method	Config	Dev chrF	Test chrF
Full FT (GPT-2 small)	$\text{lr} = 10^{-5}$, 10 ep, temp=1.2	28.077	41.199
Full FT (retrained)	fixed params, temp=1.2, len=128	27.145	–
Full FT (retrained)	fixed params, temp=0.9, len=256	26.365	–
LoRA $r = 4$ (Q+V only)	$\text{lr} = 10^{-4}$, 10 ep	incoherent	–
LoRA $r = 4$ (5 matrices)	$\text{lr} = 5 \times 10^{-6}$, 10 ep	incoherent	–
LoRA $r = 16$ (6 matrices)	$\text{lr} = 10^{-5}$, 10 ep	incoherent	–
GPT-2 Medium + LoRA $r = 16$	$\text{lr} = 10^{-4}$, 10 ep	23.948	–
Staff baseline	–	41.414	41.208

Table 3: Sonnet generation results. chrF is computed against Shakespeare reference sonnets. “Incoherent” means the model produced real Shakespeare lines followed by gibberish.

6 Analysis

6.1 LoRA Rank vs. Accuracy Tradeoff

The efficiency-accuracy tradeoff for LoRA on paraphrase detection is clear from Table 2. At $r = 2$, the model uses only 184K trainable parameters but achieves just 0.774 accuracy, a 12-point gap from full fine-tuning. At $r = 4$, accuracy jumps to 0.839 (a 6.5-point improvement over $r = 2$) while only doubling the parameter count. At $r = 8$, accuracy barely moves (0.840), suggesting the low-rank subspace is already saturated by $r = 4$.

This plateau pattern is consistent with the LoRA paper’s finding that a small rank is often sufficient for tasks where the pretrained model already has relevant representations. For paraphrase detection, the model already understands semantic similarity from pretraining; LoRA just needs to learn a small adjustment to the attention patterns to focus on the right features.

The parameter counts are: $r = 2$: 184K (0.15% of 124M), $r = 4$: 368K (0.30%), $r = 8$: 737K (0.59%). Even at $r = 8$, LoRA uses less than 0.6% of the parameters of full fine-tuning while achieving 94% of its accuracy (0.840 vs. 0.895). This is a strong efficiency result for the paraphrase task.

6.2 Why LoRA Fails on Sonnets

The sonnet results are a clear negative result, and understanding why is informative. We ran three progressively more capable LoRA configurations:

1. $r = 4$, Q+V only (2 matrices per layer): incoherent output. Likely because key and output projections are frozen, limiting the model’s ability to route information correctly.
2. $r = 4$, Q+K+V+ W_O + $W_{\text{FFN-out}}$ (5 matrices): still incoherent. The rank constraint may be too tight for the model to shift from web-text generation to Shakespearean verse.
3. $r = 16$, all 6 matrices including $W_{\text{FFN-up}}$: still incoherent. Even with more capacity, the model produces real Shakespeare lines for the first few tokens then degenerates into gibberish.

The core problem is data scarcity. With only 143 training sonnets, the model sees each sonnet roughly 10 times over 10 epochs. Full fine-tuning can memorize and generalize from this tiny corpus because all 124M parameters can shift together. LoRA, by contrast, must express the entire style shift through a low-rank bottleneck, which is insufficient when the target distribution (Shakespearean verse) is very far from the pretraining distribution (web text).

6.3 GPT-2 Medium Does Not Help

We expected GPT-2 Medium (345M parameters) with LoRA to outperform GPT-2 small with full fine-tuning, since a larger pretrained model should have richer representations. Instead, it scored 23.948 dev chrF, the worst of all configurations. Two factors explain this. First, LoRA’s rank constraint limits how much the larger model can adapt to the tiny sonnet dataset. Second, the larger model may be more prone to generating fluent but off-distribution text, since its pretraining distribution is broader and harder to shift with limited data.

6.4 chrF Metric Behavior

An important observation about the sonnet results: the chrF metric rewards character n -gram overlap with the reference sonnets, not coherence or creativity. Our best result (28.077 dev) came from the original full fine-tuning run with temperature 1.2 and max length 256. When we retrained with the same settings but slightly different generation parameters, scores dropped. When we lowered the temperature to 0.9 (which typically improves coherence), scores also dropped.

This is because lower temperature makes the model more repetitive and less likely to use rare Shakespearean vocabulary. Higher temperature and longer outputs accidentally produce more character n -gram overlap with the reference text. The test score (41.199) is much higher than the dev score (28.077) because the test set likely contains sonnets with more overlap with the training distribution. This metric quirk makes it hard to interpret the sonnet results as a true measure of generation quality.

6.5 Learning Rate Sensitivity

For full fine-tuning on paraphrase, the learning rate sweep shows that 2×10^{-5} is the sweet spot: it outperforms both 10^{-5} (0.894) and 5×10^{-5} (0.890). The lower learning rate converges more slowly but more stably; the higher rate causes some overfitting or instability in the later epochs. The improvement from tuning the learning rate is small (0.001 dev accuracy) but consistent, and the $\text{lr} = 2 \times 10^{-5}$ model achieved 0.874 test accuracy, exceeding the staff baseline of 0.856 by 1.8 points.

6.6 Error Analysis for Paraphrase Detection

The paraphrase model’s main failure mode is on borderline cases where two questions ask about the same topic but with different scope or specificity. For example, “What is the best way to lose weight?” and “How can I lose weight quickly?” are labeled as paraphrases in some contexts but not others. The cloze-style formulation may also be sensitive to the exact prompt wording; we use a fixed template and do not explore prompt variations.

Full fine-tuning achieves 0.895 dev accuracy, meaning roughly 1 in 10 predictions is wrong. LoRA at $r = 4$ achieves 0.839, meaning roughly 1 in 6 is wrong. The gap is likely concentrated on these borderline cases, where the model needs finer-grained representations that LoRA’s low-rank update cannot fully capture.

7 Conclusion

We built GPT-2 from scratch and investigated full fine-tuning vs. LoRA across two data regimes. The main findings are:

1. **LoRA works well on large datasets.** On paraphrase detection (141K examples), LoRA $r = 4$ achieves 0.839 accuracy using only 0.30% of the parameters of full fine-tuning (0.895). The rank-accuracy curve plateaus at $r = 8$, suggesting the relevant subspace has low intrinsic dimensionality.
2. **LoRA struggles on tiny datasets.** On sonnet generation (143 training examples), all LoRA configurations produced incoherent output. The rank constraint prevents the model from shifting far enough from its pretraining distribution.

3. **Hyperparameter tuning matters.** A simple learning rate sweep on paraphrase detection found 2×10^{-5} as the sweet spot, improving dev accuracy from 0.894 to 0.895 and test accuracy to 0.874 (vs. staff baseline 0.856).
4. **Model scaling does not always help.** GPT-2 Medium with LoRA $r = 16$ underperformed GPT-2 small with full fine-tuning on the tiny sonnet dataset, suggesting that larger models need more data to leverage their additional capacity.
5. **Evaluation metrics can be misleading.** The chrF metric for sonnets rewards character n -gram overlap rather than coherence, making it hard to distinguish good generation from lucky vocabulary overlap.

The primary limitation of this work is the tiny sonnet dataset, which makes it hard to draw strong conclusions about LoRA for generation. Future work could explore data augmentation (e.g., using other Shakespeare plays), prefix tuning, or a retrieval-augmented approach that conditions generation on relevant training sonnets. For paraphrase detection, exploring higher LoRA ranks or applying LoRA to the FFN up-projection might close the remaining gap with full fine-tuning.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- [4] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799, 2019.
- [5] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4582–4597, 2021.
- [6] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.
- [7] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs. <https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs>, 2017.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.
- [9] Tim van de Cruys. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2471–2480, 2020.
- [10] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- [11] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

A Appendix

A.1 LoRA Parameter Counts

Configuration	Trainable Params	% of 124M
Full fine-tuning	124,439,808	100.0%
LoRA $r = 2$ (5 matrices, 12 layers)	184,320	0.15%
LoRA $r = 4$ (5 matrices, 12 layers)	368,640	0.30%
LoRA $r = 8$ (5 matrices, 12 layers)	737,280	0.59%
LoRA $r = 16$ (6 matrices, 12 layers)	1,769,472	1.42%

Table 4: Trainable parameter counts for each configuration. The 5-matrix configuration covers $W_Q, W_K, W_V, W_O, W_{\text{FFN-out}}$; the 6-matrix configuration also includes $W_{\text{FFN-up}}$.

A.2 Training Curves

For paraphrase detection, the training loss decreases monotonically across all configurations. Full fine-tuning with $\text{lr} = 10^{-5}$: epoch losses 0.365, 0.272, 0.222, 0.181, 0.146. The corresponding dev accuracies are 0.867, 0.882, 0.892, 0.893, 0.894, showing steady improvement with no sign of overfitting over 5 epochs.

For sonnet generation, the full fine-tuning baseline reaches train loss 3.877 at epoch 9. The loss curve decreases steadily, but the absolute loss value is high because the model is predicting from a 50,257-token vocabulary with a very small training set.

A.3 Sample Sonnet Output

Below is a representative excerpt from the full fine-tuning baseline ($\text{temp}=1.2$, $\text{max_length}=256$):

*Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date...*

The model successfully reproduces Shakespearean vocabulary and iambic pentameter for the opening lines. Later in the generated sonnet, coherence degrades as the model moves further from the training distribution. This pattern is consistent across all configurations and explains why longer outputs ($\text{max_length}=256$) score better on chrF than shorter ones: the first few lines are high quality and contribute substantial character n -gram overlap.