

Cost–Performance Tradeoffs for GPT-2 Fine-Tuning: A Case Study on Paraphrase and Sonnet Continuation

Stanford CS224N Default Project

Ricardo Ruiz

Stanford University

`rruiz26@stanford.edu`

TA mentor: Advit Deepak

External collaborator, Mentor: No, No

Sharing project: No

Late days: 1

Abstract

Fine-tuning language models is often treated as a straightforward way to improve downstream performance, but in practice, it is expensive (both in time and compute) and sensitive to many design choices. In this project, we study the performance-cost tradeoff of adapting GPT-2 to two downstream tasks: Quora paraphrase detection (accuracy) and Shakespeare sonnet continuation (chrF), measuring cost primarily in GPU-hours on T4s and A100s (used when models do not fit on T4). Our focus is practical and exploratory rather than theoretical, and we use these tasks to extract a few stylized facts about what tends to matter most under a fixed budget.

First, task-specific changes can rival or beat brute-force scaling. For paraphrase detection, symmetry-aware data augmentation improves dev accuracy, and our best GPT-2 Medium full fine-tuned model reaches 0.912 dev accuracy, matching or exceeding larger LoRA-based configurations in our runs. Second, for sonnet continuation, inference-time decoding can dominate performance even when training is held fixed: on our best checkpoint, top-k and nucleus sampling outperform greedy decoding and beam search by a large margin. However, the paraphrase results also reveal a substantial dev–test gap, underscoring the importance of evaluating cost-aware improvements together with generalization. Together, these findings suggest that the best “bang for buck” is task-dependent, and that improving the experimental baseline (e.g., through better data preparation or decoding strategy) can be more cost-effective than scaling model size.

1 Introduction

Full fine-tuning is the standard way to adapt pretrained language models to downstream tasks, but it is often unclear which modifications are actually worth the compute. Even at GPT-2 scale, small changes in training data construction, decoding, or parameter-updating strategy can substantially affect both performance and runtime.

Compute is a real constraint: it costs money, takes wall-clock time, and has an environmental footprint. Prior work has argued that efficiency and pricetags should be reported alongside accuracy [1]. In our experiments, this issue appears even for GPT-2: training runs can take many hours to days, and comparing even a modest number of variants becomes expensive and a bottleneck to experimentation.

This motivates a cost-aware case study. Instead of chasing a single best score, we measure the tradeoff between compute and performance for a small set of common interventions on two CS224N default-project extension tasks: Quora paraphrase detection (accuracy) and Shakespeare sonnet continuation (chrF). We log GPU-hours and estimated dollar cost, using T4 GPUs by default and A100 GPUs only when a configuration does not fit on T4.

Our results support two stylized facts. First, task-specific changes can rival or beat brute-force scaling. For paraphrase detection, symmetry-aware swap augmentation improves GPT-2 Medium + LoRA from 0.895 to 0.904 dev accuracy, and the best GPT-2 Medium full fine-tuned model reaches 0.912 dev accuracy, matching or exceeding larger LoRA-based configurations. Second, for sonnet continuation, inference-time decoding can dominate performance even when training is held fixed: on our best checkpoint, chrF varies from 25.99 to 42.37 depending only on the decoding strategy. At the same time, the paraphrase results reveal a substantial dev–test gap, showing that cost-aware improvements still need to be evaluated for generalization. Together, these findings suggest that improving the experimental baseline can be more cost-effective than scaling model size.

Summary of contributions.

- We evaluate training-time and inference-time ablations for paraphrase detection and sonnet continuation, including symmetry-based data augmentation, LoRA scaling, domain-adaptive transfer learning for sonnets, rhyme/structure-tag variants, and decoding constraints.
- We report task-dependent cost–performance tradeoffs and analyze failure modes that help explain when targeted modifications outperform brute-force scaling.

2 Related Work

Our work is related to three areas of research: compute-aware evaluation in NLP, parameter-efficient fine-tuning, and decoding for language generation.

Compute-aware NLP: Recent work argues that NLP systems should be evaluated not only by accuracy, but also by the resources required to obtain it. (author?) [1] advocate treating efficiency as a first-class criterion, while (author?) [2] quantify the financial and environmental costs of training modern NLP systems. We follow this perspective by reporting performance together with GPU-hours and estimated dollar cost.

Parameter-efficient fine-tuning: Parameter-efficient fine-tuning methods reduce the cost of adapting large models by updating only a subset of parameters or by adding small task-specific modules. Examples include adapters [3], prefix tuning [4], and LoRA [5], which learns low-rank updates to frozen weight matrices. QLoRA [6] further combines quantization with LoRA to reduce memory use. In our work, we use LoRA as our main scaling method and compare it against full fine-tuning and task-aware interventions such as data augmentation.

Decoding for generation: For text generation, inference-time decoding can substantially change output quality even when the trained model is fixed Holzman Et al. (author?) [7] show that the quality of decoding strategies greatly affect machine text, and propose nucleus sampling as a good decoding strategy. In our sonnet experiments, decoding strategy changes chrF substantially for any

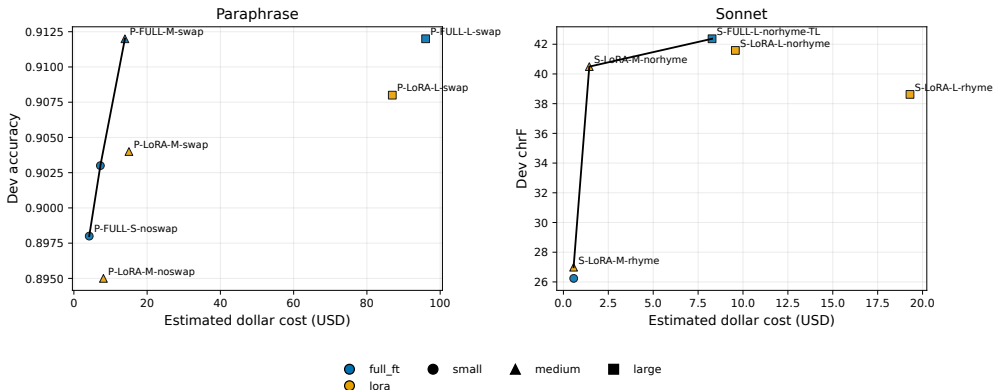


Figure 1: Cost–performance tradeoff for paraphrase detection and sonnet continuation.

given checkpoint, which reinforces the idea that decoding should be treated as part of the overall system rather than as a fixed post-processing step.

3 Approach

Our goal is not to propose a new training algorithm, but to systematically study the performance-cost tradeoff of several common modifications when fine-tuning GPT-2 on downstream tasks. We group our methods into three categories: (1) task-aware data modifications, (2) parameter-efficient scaling methods, and (3) inference-time methods. For each method, we measure both task performance and compute cost (GPU-hours). We focus on interventions that are broadly applicable across standard GPT-2 fine-tuning setups, rather than hardware-specific optimizations such as FlashAttention [8].

3.1 Baseline Model

Our baseline system uses the provided CS224N GPT-2 implementation and training pipeline, initialized from pretrained GPT-2 checkpoints. Given an input sequence x , the model defines the standard causal language modeling distribution

$$p(x) = \prod_{t=1}^T p(x_t \mid x_{<t}).$$

For paraphrase detection, we follow the default project and cast the task as a cloze-style prediction problem: the input consists of two questions and a prompt asking whether they are paraphrases, and the model predicts the token corresponding to “yes” or “no.” For sonnet continuation, the model receives the first three lines of a Shakespeare sonnet and autoregressively generates the remaining lines.

Unless otherwise stated, all modifications described below were implemented by us on top of the provided project code.

3.2 Cost Measurement

To evaluate cost efficiency, we log wall-clock training time and compute total GPU-hours for each experiment:

$$\text{GPU-hours} = (\text{wall-clock hours}) \times (\text{number of GPUs}).$$

Most experiments are run on NVIDIA T4 GPUs. When a configuration did not fit in T4 memory, we used A100 GPUs instead. This gives us a relatively consistent hardware setup while still allowing us to study larger models. For dollar-cost estimates, we use approximate GCP us-west1 rental prices of \$0.35 per GPU-hour for T4 and \$3.67 per GPU-hour for A100 instances, so A100 training is roughly 10 times more expensive. We should note that in principle, some runs that fit on T4 could still be cheaper on A100 due to higher throughput, but we do not systematically explore that tradeoff here because it depends strongly on architecture-specific kernels and we were constrained by cloud availability of A100s.

3.3 Task-Aware Data Modifications

Symmetry augmentation for paraphrase. The Quora paraphrase task is symmetric: if (A, B) is labeled as a paraphrase pair, then (B, A) should receive the same label. However, the dataset contains only one ordering for each pair. To encourage the model to learn this invariance, we augment the training data with additional swapped pairs:

$$(A, B) \rightarrow y \iff (B, A) \rightarrow y.$$

This modification does not change the architecture, but it doubles the number of training examples and encourages the model to learn the invariance.

Structural tags for sonnets. For sonnet generation, we also experiment with explicit structural tags intended to expose the sonnet’s poetic form to the model. In particular, we add tokens indicating stanza and rhyme-group identity. This lets us test whether making poetic structure explicit helps the model generate continuations that better match the reference sonnets.

3.4 Parameter-Efficient Fine-Tuning

To explore scaling under limited compute budgets, we evaluate Low-Rank Adaptation (LoRA) [5]. Instead of updating a full weight matrix W , LoRA learns a low-rank update

$$W' = W + BA,$$

where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ with rank $r \ll \min(d, k)$. The pretrained weights W remain frozen while only the low-rank matrices are trained. This reduces the number of trainable parameters and allows larger models to be fine-tuned under tighter memory constraints.

We compare LoRA against full fine-tuning on GPT-2 Medium and GPT-2 Large. We do not emphasize LoRA results on GPT-2 Small because, in our setup, LoRA often increased training time relative to simply fine-tuning the full small model. This is due to the mechanics of backpropagation.

3.5 Inference-Time Methods

In addition to training-time modifications, we study several inference-time methods that do not require retraining but can still substantially affect downstream performance.

Decoding strategies for sonnet continuation. For the sonnet task, we evaluate multiple decoding strategies. Given the model distribution $p(x_t \mid x_{<t})$, greedy decoding selects the most probable token at each step, while stochastic decoding methods sample from a restricted subset of tokens.

Top- k sampling restricts the candidate set to the k most probable tokens and renormalizes the distribution. Nucleus sampling selects the smallest token set whose cumulative probability exceeds a threshold p [7]. We also evaluate beam search. Across these methods, we vary decoding hyperparameters such as k , temperature, and nucleus threshold in order to measure how sensitive chrF is to inference-time choices.

Order ensembling for paraphrase detection. For paraphrase detection, we also experiment with an inference-time symmetry heuristic. Given a pair of questions (A, B) , we evaluate the model on both (A, B) and (B, A) and then aggregate the two predictions. In our implementation, we predict paraphrase only if both orderings predict paraphrase. This provides a simple way to enforce symmetry at inference time without retraining the model. There are many other ways to aggregate, but due to time constraints, we did not explore this path further.

3.6 Alternative Training Procedure for Sonnets

Finally, for sonnet generation, we experiment with a two-stage transfer/domain learning setup. We first continue fine-tuning on a non-Shakespeare sonnet corpus drawn from the Poetry Foundation dataset [9], after filtering the data to retain sonnets and remove Shakespeare poems. We then fine-tune the resulting checkpoint on the Shakespeare sonnet training split. The goal is to expose the model to broader poetic structures before specializing in Shakespearean sonnets.

4 Experiments

This section describes the datasets, evaluation metrics, and experimental setup used in our study, and then reports the quantitative results for the main configurations we evaluated. Unless otherwise stated, the tables below report the *best checkpoint reached* for each configuration, together with the GPU-hours and estimated dollar cost required to reach that checkpoint.

Model	Method	Swap Data	Inference	Dev Acc.	GPU-hrs	Est. Cost (\$)
GPT-2 Small	Full FT	No	Default	0.898	11.98	4.19
GPT-2 Small	Full FT	Yes	Default	0.903	20.67	7.23
GPT-2 Medium	LoRA	No	Default	0.895	23.05	8.07
GPT-2 Medium	LoRA	Yes	Order ensemble	0.904	42.96	15.04
GPT-2 Medium	Full FT	Yes	Order ensemble	0.912	3.80	13.94
GPT-2 Large	LoRA	Yes	Order ensemble	0.908	23.69	86.93
GPT-2 Large	Full FT	Yes	Order ensemble	0.912	26.17	96.03

Table 1: Paraphrase detection results at the best checkpoint reached for each configuration. “Swap Data” indicates symmetry-based augmentation where both (A, B) and (B, A) are included during training. Estimated cost uses \$0.35/GPU-hour for T4 and \$3.67/GPU-hour for A100.

Configuration	Swap Data	Inference	Dev Acc.
Small Full FT baseline	No	Default	0.898
Small Full FT + order ensemble	No	Order ensemble	0.894

Table 2: Order ensembling alone slightly hurts paraphrase accuracy when the model is not trained on swapped pairs.

4.1 Data

We evaluate on the two CS224N default-project extension tasks: Quora paraphrase detection and Shakespeare sonnet continuation. For transfer-learning experiments on sonnet generation, we additionally use a filtered subset of the Poetry Foundation dataset [9] restricted to non-Shakespeare sonnets.

4.2 Evaluation Method

We report accuracy for paraphrase detection and chrF [10] for sonnet continuation, using both dev-set results and official leaderboard test results for the final submitted checkpoints.

4.3 Experimental Details

All experiments start from pretrained GPT-2 checkpoints. We evaluate GPT-2 Small (124M parameters), GPT-2 Medium (355M), and GPT-2 Large (774M). Full fine-tuning runs use the provided training pipeline, while parameter-efficient experiments use LoRA [5].

Most runs were executed on NVIDIA T4 GPUs. When a model did not fit on T4, we used A100 GPUs. We measured compute in GPU-hours and converted this to an approximate rental cost using \$0.35/GPU-hour for T4 and \$3.67/GPU-hour for A100-class instances. For paraphrase, most full fine-tuning and LoRA runs use a learning rate of 1×10^{-5} , with a few additional variants at 2×10^{-5} . For sonnet generation, LoRA runs use either 3×10^{-5} or 1×10^{-4} depending on the configuration.

4.4 Results

Paraphrase Detection. Table 1 summarizes the main paraphrase configurations. Three patterns are immediate. First, swap augmentation helps: for example, GPT-2 Medium + LoRA improves from 0.895 to 0.904 when trained with swapped pairs. Second, the best overall validation result is achieved by GPT-2 Medium with full fine-tuning and swapped data, which reaches 0.912 dev accuracy. Third, larger models do not always dominate smaller ones: on the development set, the best GPT-2 Medium full fine-tuned model matches the best GPT-2 Large full fine-tuned model while requiring substantially less compute and cost.

To isolate the effect of inference-time order ensembling alone, Table 2 compares two small-model runs without swap augmentation. In this setting, requiring both (A, B) and (B, A) to predict paraphrase slightly reduces accuracy, suggesting that order ensembling is not beneficial on its own.

Sonnet Continuation. Table 3 summarizes the best sonnet-generation checkpoints. Two patterns stand out. First, removing rhyme tags consistently benefits both medium and large LoRA models: they perform substantially better without them. Second, the best sonnet result comes from GPT-2

Model	Method	Training Variant	Best Decoding	Dev chrF	GPU-hrs	Est. Cost (\$)
GPT-2 Small	Full FT	Default	Default	16.28	0.34	0.12
GPT-2 Small	Full FT	No rhyme tags	Best decode	26.24	1.64	0.57
GPT-2 Medium	LoRA	Rhyme tags	Best decode	26.99	1.62	0.57
GPT-2 Medium	LoRA	No rhyme tags	Best decode	40.49	4.12	1.44
GPT-2 Medium	LoRA	No rhyme tags ($\eta = 10^{-4}$)	Best decode	40.95	7.00	2.45
GPT-2 Large	LoRA	Rhyme tags	Best decode	38.62	5.26	19.30
GPT-2 Large	LoRA	No rhyme tags	Best decode	41.58	2.61	9.58
GPT-2 Large	Full FT	No rhyme tags + transfer learning	Top- k	42.37	2.26	8.28

Table 3: Sonnet continuation results at the best checkpoint reached for each configuration. “Best decode” means the strongest decoding strategy evaluated for that checkpoint. Transfer learning refers to continued fine-tuning on non-Shakespeare sonnets before fine-tuning on the Shakespeare training split.

Decoding Strategy	Dev chrF
Greedy	37.97
Top- k ($k = 20$)	41.77
Top- k ($k = 40$)	42.37
Top- k ($k = 80$)	42.00
Nucleus ($T = 1.0, p = 0.95$)	42.23
Beam-2	32.26
Beam-4	27.81
Beam-8	25.99

Table 4: Decoding sensitivity on the best sonnet checkpoint (GPT-2 Large, full fine-tuning with transfer learning, no rhyme tags).

Large with full fine-tuning and transfer learning on non-Shakespeare sonnets, achieving 42.37 dev chrF.

Because sonnet quality is highly sensitive to inference, we also report decoding results for the best checkpoint in Table 4. On this single checkpoint, chrF ranges from 25.99 to 42.37 depending only on the decoding strategy. In particular, top- k and nucleus sampling perform best, while beam search consistently performs much worse.

4.5 Leaderboard Test Results

Tables 5 and 6 report the official leaderboard test results for the final checkpoints we submitted. These results are informative because they reveal different generalization behavior across the two tasks.

Paraphrase Detection. Although several paraphrase models achieved strong dev accuracy, their test performance was much lower and tightly clustered. In particular, the best dev model (GPT-2 Medium, full fine-tuning with swapped data) reached 0.912 dev accuracy but only 0.856 on the held-out test set. A similar gap appears for the larger full and LoRA models. This suggests that the modifications that helped most on dev did not transfer as strongly to the test distribution.

Sonnet Continuation. For sonnet generation, the test results are more consistent with the dev trends. The best test result comes from GPT-2 Large with full fine-tuning, no rhyme tags, transfer learning, and nucleus decoding, which reaches 42.08 chrF. Transfer learning appears to help modestly on the test set as well: the transfer-learned model outperforms the otherwise similar large full fine-tuned model without transfer learning.

Hyperparameter Variation. We also explored a small number of hyperparameter changes, especially for sonnet generation and large-model paraphrase runs. These variations occasionally improved performance slightly, but the gains were generally smaller than those obtained from task-aware modifications such as swap augmentation or decoding changes. For this reason, we treat hyperparameter tuning as secondary to the structural interventions studied in this paper, but it should be noted that given the cost of hyperparameter tuning in our setup we got more value out of modeling changes than tuning changes.

Submitted Model	Dev Acc.	Test Acc.
GPT-2 Medium, Full FT, Swap	0.912	0.856
GPT-2 Large, Full FT, Swap	0.912	0.855
GPT-2 Large, LoRA, Swap	0.908	0.855

Table 5: Official test leaderboard results for paraphrase detection. Despite strong dev performance, all three submitted models show a substantial dev–test gap.

Submitted Model + Decoding	Dev chrF	Test chrF
Large Full FT + TL + Top- k ($k = 40$)	42.37	41.41
Large Full FT + TL + Nucleus ($T = 1.0, p = 0.9$)	42.08	42.08
Large Full FT (no TL) + Nucleus ($T = 1.0, p = 0.9$)	41.41	41.41

Table 6: Official test leaderboard results for sonnet continuation. Transfer learning remains helpful on the test set, though the best decoding strategy differs slightly between dev and test.

5 Analysis

We now analyze why certain interventions helped and where they failed to generalize. Our main findings are that (i) task-aware changes can strongly improve dev performance, but do not always transfer to the hidden test distribution, and (ii) for generation, inference-time decisions can matter as much as or more than training-time scaling.

5.1 Paraphrase: Symmetry Helps on Dev, But Not on Test

The Quora paraphrase task is symmetric: if (A, B) is a paraphrase pair, then (B, A) should receive the same label. Augmenting the training data with swapped pairs improves dev accuracy consistently; for example, GPT-2 Medium + LoRA improves from 0.895 to 0.904, and the best GPT-2 Medium full fine-tuned model reaches 0.912 dev accuracy.

However, these gains do not transfer nearly as strongly to the hidden test set. The best dev model (GPT-2 Medium, full fine-tuning, swapped data) reaches only 0.856 test accuracy, and the larger full and LoRA models are nearly identical on test. This suggests either overfitting to the train/dev distribution or a distribution mismatch between the provided dev set and the hidden test set. We did not identify an obvious implementation bug, so a more detailed error analysis would be needed to distinguish between these possibilities.

5.2 Sonnets: Decoding and Tags Matter More Than Scaling

For sonnet continuation, decoding strategy has a large impact even when the trained checkpoint is fixed. On our best checkpoint, dev chrF ranges from 25.99 to 42.37 depending only on decoding choice, with top- k and nucleus sampling performing best and beam search performing much worse. In our experiments, this decoding-induced variation is larger than the gain from increasing model size.

We also find that explicit rhyme and structural tags consistently hurt performance. For example, GPT-2 Medium + LoRA achieves 40.49 chrF without tags but only 26.99 with them; the same pattern holds for GPT-2 Large. One plausible explanation is that these tags shift the training distribution and force the model to model artificial tokens rather than the language of the sonnets themselves. By contrast, transfer learning on non-Shakespeare sonnets provides a modest but consistent gain and produces the best test result. We wish to further explore how adding tokens affect downstream performance in the future as it seems to be significant and detrimental.

5.3 Cost–Performance Lessons

Taken together, our results suggest that the best “bang for buck” is task-dependent. For paraphrase detection, the most effective improvement was a task-aware data modification (swap augmentation), not model scaling. For sonnet generation, the largest gains came from inference-time decoding choices and from avoiding harmful structural tags. Larger models sometimes improved performance, but these improvements were not uniform and often came at substantially higher cost. Overall, our

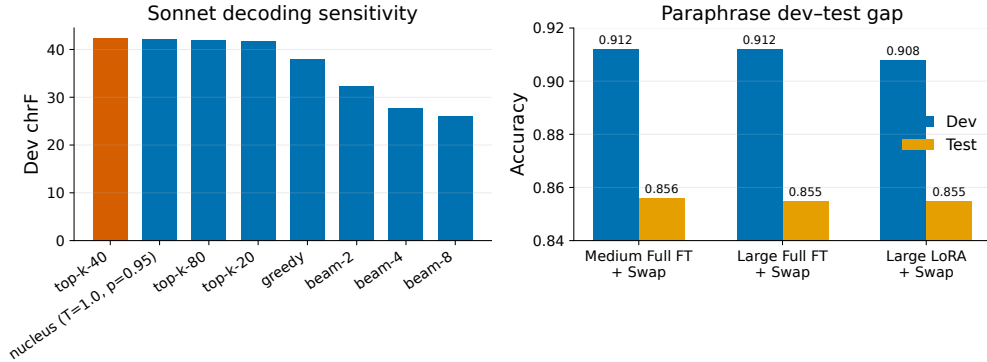


Figure 2: Two key analysis findings. Left: decoding strategy strongly affects sonnet continuation quality on the same checkpoint, with top- k and nucleus sampling substantially outperforming beam search. Left: An example with small dispersion relative to other checkpoints Right: paraphrase models with strong dev accuracy generalize much more weakly on the hidden test set, producing a large dev-test gap.

experiments suggest that improving the experimental baseline—through data construction, decoding, or other problem-aware changes—can be more cost-effective than scaling model size alone.

6 Conclusion

In this project, we studied the performance–cost tradeoff of adapting GPT-2 to Quora paraphrase detection and Shakespeare sonnet continuation. Rather than focusing only on the best final metric, we tracked the compute required to obtain each improvement and compared training-time, inference-time, and model-scaling strategies.

Our results suggest three main lessons. First, task-aware modifications can be highly effective: for paraphrase detection, symmetry-based swap augmentation consistently improved dev accuracy, and a GPT-2 Medium full fine-tuned model with swapped data achieved the best dev result in our experiments. Second, inference-time choices can matter as much as training-time changes for generation. For sonnet continuation, the same checkpoint produced dev chrF scores ranging from 25.99 to 42.37 depending only on the decoding strategy, with top- k and nucleus sampling clearly outperforming beam search in all of the checkpoints that we evaluated. Third, larger models did not uniformly dominate smaller ones once compute and costs were taken into account.

At the same time, our results highlight an important limitation of dev-set conclusions. In paraphrase detection, the strongest dev models generalized much more poorly than expected on the hidden test set, suggesting either overfitting to the train/dev distribution or a dev-test distribution mismatch. By contrast, the sonnet results transferred more consistently, and the best test result came from GPT-2 Large with full fine-tuning, no rhyme tags, transfer learning, and tuned decoding.

Overall, we find that improving the experimental baseline, through task-aware data augmentation, transfer learning, or careful decoding, can sometimes rival or exceed the gains from brute-force scaling. These results reinforce the value of understanding the task and evaluation pipeline before investing in larger models or more expensive optimization.

Practical Takeaway. Before scaling model size or running expensive hyperparameter sweeps, improve the experimental baseline first. In our experiments, task-aware data augmentation and decoding choices often delivered larger gains per unit of compute than moving to a larger GPT-2 variant.

References

- [1] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *arXiv preprint arXiv:1907.10597*, 2019.

- [2] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.
- [3] 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 (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.
- [4] 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 and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.
- [5] 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 (ICLR)*, 2022.
- [6] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- [7] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020.
- [8] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [9] Tushar Goyal. Poetry foundation poems dataset. <https://www.kaggle.com/datasets/tgdivy/poetry-foundation-poems>, 2018. Data scraped from PoetryFoundation.org.
- [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, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

7 Appendix

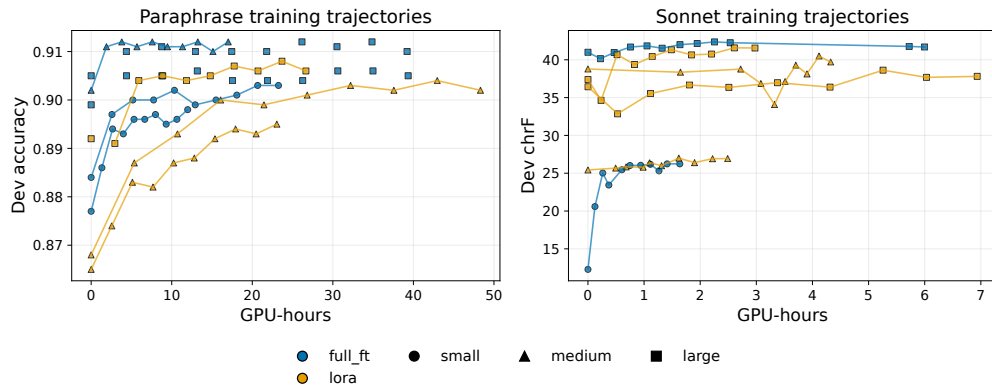


Figure 3: Dev accuracy evolution over time

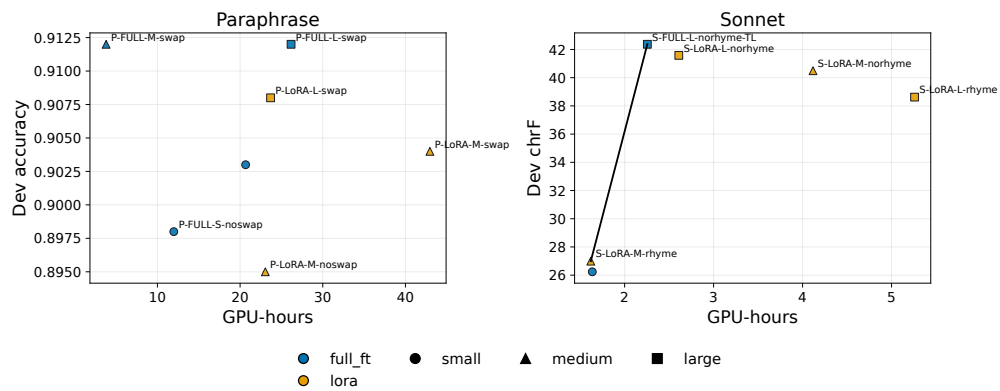


Figure 4: Hours vs performance