

Modernizing GPT-2: Integrating Low-Rank Adaptation, FlashAttention, and Multi-Token Prediction for Efficient Sonnet Generation

Stanford CS224N {Default}

Manan Sheth

Department of Computer Science
Stanford University
manan1@stanford.edu

Sanjay Dixit Bhuvanagiri

Department of Computer Science
Stanford University
sanjaydb@stanford.edu

Abstract

We investigate how several modern architectural and training techniques can be incorporated into a GPT-2–style language model and evaluate their impact across three tasks: sentiment classification, paraphrase detection, and structured sonnet generation. Starting from a GPT-2 baseline implementation, we integrate parameter-efficient fine-tuning with Low-Rank Adaptation (LoRA), memory-efficient attention using FlashAttention-2, contrastive representation learning inspired by SimCSE, and Multi-Token Prediction (MTP) training objective motivated by DeepSeek-V3. Our experiments show that parameter-efficient scaling from GPT-2 Base (117M) to GPT-2 XL (1.5B) using LoRA and FlashAttention-2 improves paraphrase detection accuracy on Quora Question Pairs from 0.897 to 0.903 on the development set and achieves 0.901 on the test set. For sentiment classification, we find that full-model fine-tuning is highly sensitive to the learning rate, whereas last-linear-layer tuning provides more stable optimization across settings. In structured sonnet generation, Multi-Token Prediction (MTP) improves chrF scores for larger models, indicating that predicting multiple future tokens helps capture short-range structural dependencies in poetic text.

1 Key Information to include

- TA mentor: ahmedah@cs.stanford.edu
- External collaborators (if no, indicate “No”): No
- External mentor (if no, indicate “No”): No
- Sharing project (if no, indicate “No”): No
- Late Days (if you are using any):

Team member	Late days remaining
Manan Sheth	5
Sanjay Dixit Bhuvanagiri	2
Total pooled	7
Team size	2
Late days applied for the report	4 (2 each for a total of 2 days)

2 Introduction

Large Language Models (LLMs) are evolving so quickly since the introduction of GPT-2. New ideas in architectures rely on massive parameter counts, memory-efficient attention kernels, and

sophisticated training objectives to achieve state-of-the-art reasoning and generation. While these innovations have driven substantial progress in modern large-scale models, many of them have been designed for newly developed architectures and large industrial training environments. As a result, it remains unclear how effectively these techniques can be applied to earlier autoregressive models such as GPT-2, particularly under the computational constraints typical of academic settings. We studied whether several modern training and efficiency techniques can improve a GPT-2 baseline even under these constraints.

We systematically tried applying these enhancements to three tasks: Sentiment Analysis (SST, CFIMDB), Paraphrase Detection (Quora), and Sonnet Generation. When we tried scaling the full-model fine-tuning to billion-parameter it proved out to be computationally prohibitive on standard hardware we used from Google Cloud Platform. Therefore, we integrate Low-Rank Adaptation (LoRA) [1] and FlashAttention-2 [2] to train a GPT-2 XL (1.5B) model efficiently. Also, we saw that generative models did really struggle with semantic tasks like paraphrase detection, we implemented SimCSE contrastive learning [3]. To address the lack of structural "planning" in autoregressive generation, we implemented Multi-Token Prediction (MTP) [4].

3 Related Work

Parameter-Efficient Fine-Tuning: Fine-tuning all parameters of modern LLMs has become computationally hard and less straightforward. Paper from Hu et al. [1] mentions that deploying independent instances of fully fine-tuned models is economically and practically very expensive for most applications. Low-Rank Adaptation (LoRA) is helpful in such situation by freezing pre-trained weights and injecting trainable rank decomposition matrices into the transformer layers. This mathematically exploits the low intrinsic dimension of over-parameterized models, reducing the trainable parameter footprint by up to $10,000\times$ and GPU memory requirements by $3\times$ compared to standard fine-tuning. Also, unlike adapter layers, LoRA introduces zero additional inference latency because low-rank matrices can be merged with the frozen weights for the deployment.

Efficient Attention: Scaling Transformers to longer sequence lengths is constrained due to attention layer, as its runtime and memory increase quadratically $O(N^2)$ with sequence length. Also the standard Multi-Head Attention is bottlenecked heavily by High Bandwidth Memory (HBM) read/write operations. Paper by Dao [2] introduced FlashAttention-2 to optimize this by exploiting the asymmetric GPU memory hierarchy. By improving work partitioning between thread blocks and warps on the GPU, it reduces non-matmul FLOPs and parallelizes attention computation over the sequence length dimension. This idea prevents the materialization of the large attention matrix in HBM, addressing low-occupancy and unnecessary shared memory reads/writes, and ultimately reaching up to 73% of the theoretical maximum FLOPs/s.

Contrastive Representation Learning: The paper SimCSE [3] helps us in advancing the sentence embedding quality by utilizing contrastive learning frameworks. By pulling semantically equivalent sentences closer and pushing disparate ones apart in the latent space, it significantly improves causal language models on semantic textual similarity and paraphrase tasks, and so basically compensating for the lack of bidirectional self-attention found in encoder-only models.

Advanced Training Objectives: Most of the traditional LLMs were using only Next-Token Prediction (NTP). Then recently DeepSeek-V3 [4], showed that Multi-Token Prediction (MTP)—where the model predicts multiple future tokens at each position helps in densifying the training signal and encourages the model to "plan" its outputs. DeepSeek-V3 utilizes this objective and also the auxiliary-loss-free strategy for load balancing. We use the MTP objective as an auxiliary loss, which proves effective for structured text generation and data efficiency.

4 Approach

Our base architecture is a GPT-2 transformer implemented from scratch, which we extended with the following modern modules:

4.1 Low-Rank Adaptation (LoRA)

To scale to the GPT-2 XL model (1.5 Billion parameters) on limited GPU hardware (e.g., NVIDIA L4 24GB that was used for most of our training runs), we implemented LoRA. We targeted the attention projection matrices (query, key, value) as well as the MLP dense layers (intermediate, output). For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, LoRA constrains the update by representing it with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and $\text{rank } r \ll \min(d, k)$.

As per the LoRA paper from Hu et al. [1], we initialized A with a random Gaussian distribution and B with zeros, making sure that $\Delta W = 0$ at the start of training. We utilized a rank $r = 8$ and a scaling factor $\alpha = 16$. This configuration reduced the trainable parameters on GPT-2 XL to exactly 9,833,602, representing only tiny 0.626% of the 1.57B total parameters. This massive reduction in state allowed us to fit the optimizer and gradients in limited memory.

4.2 FlashAttention-2 via Scaled Dot-Product Attention

To optimize our multi-head attention mechanism, we changed from standard manual matrix multiplications with PyTorch’s hardware-optimized `scaled_dot_product_attention`, which dispatches to FlashAttention-2 backend algorithms natively. We explicitly combined causal masking with sequence padding masks, replacing padded tokens with -1×10^4 prior to the softmax computation, bypassing the $O(N^2)$ VRAM instantiation of dense attention matrices.

4.3 Contrastive Learning (SimCSE)

For the Quora Paraphrase Detection task, standard cross-entropy loss over a linear classification head often fails to maximize the margin between positive and negative semantic pairs. We augmented the classification objective with a SimCSE-style contrastive loss. Given representations h_1 and h_2 for a positive paraphrase pair within a batched subset, we compute cosine similarities across the batch:

$$\mathcal{L}_{\text{contrastive}} = -\log \frac{e^{\text{sim}(h_1, h_2)/\tau}}{\sum_j e^{\text{sim}(h_1, h_j)/\tau}}$$

We had set temperature hyperparameter $\tau = 0.05$ to give penalty to hard negatives, and we weighted the contrastive loss component at $\lambda = 0.1$ relative to the main classification loss for a stable multi-task optimization.

4.4 Multi-Token Prediction (MTP)

To improve the structure of the poem in Shakespearean sonnets, we modified the autoregressive objective. Standard GPT-2 minimizes the cross-entropy of predicting token x_{t+1} given h_t . Inspired by the auxiliary-loss strategy of DeepSeek-V3 [4], we implemented a simplified MTP strategy as mentioned below which computes loss for token x_{t+k} at varying horizons:

$$\mathcal{L}_{MTP} = \sum_{k=1}^K \lambda_k \mathcal{L}_{CE}(P(x_{t+k}|h_t), x_{t+k})$$

We implemented architectures supporting both `singlehead` (sharing the language modeling head) and `multihead` (separate linear projections for each future step) possibilities with configurable horizon weighting. By forcing the model to predict $t + 2$ and beyond, we added penalties for long-term structural drift during poetic generation, effectively enforcing rhythmic planning in the latent representations.

5 Experiments

5.1 Data

- **Sentiment Analysis:** We evaluated on the Stanford Sentiment Treebank (SST) (5 classes) and the CFIMDB dataset (binary movie reviews).

- **Paraphrase Detection:** We utilized the Quora Question Pairs dataset ($\sim 140k$ training examples).
- **Sonnet Generation:** We used the provided corpus of 154 Shakespearean sonnets.

5.2 Evaluation method

We evaluate our models across three distinct tasks, utilizing metrics tailored to each specific domain:

Sentiment Analysis: For the Stanford Sentiment Treebank (SST) and CFIMDB datasets, we formulated sentiment prediction as a cloze-style classification task. We evaluated model performance primarily using accuracy, that measures the proportion of correctly predicted sentiment classes (5 classes for SST, binary for CFIMDB).

Paraphrase Detection: On Quora Question Pairs dataset, we evaluated the model’s ability to identify semantic equivalence between pairs of sentences. Performance is measured using standard classification accuracy on the held out development and test sets.

Sonnet Generation: To evaluate the structural coherence and linguistic similarity of our generated text to the reference Shakespearean corpus, we utilized the **chrF score** (character n-gram F-score) [5]. This metric computes character-level n-gram overlaps, making it very effective for evaluating the stylistic, morphological, and rhythmic features inherent to structured poetic generation.

5.3 Experimental Details

Models were optimized using (our implementation of) the AdamW optimizer with a batch size of 8 by default. For stability on larger variants, we used mixed-precision training (AMP autocast) and gradient accumulation up to 4 steps for Sentiment Analysis and Paraphrase Detection tasks. We heavily tuned the learning rate (lr), testing 1×10^{-3} and 1×10^{-5} based on the fine-tuning paradigm (Last-Linear Layer vs. Full Model). Since we used Google Colab free version and Google Cloud preemptible instances time to time we had added checkpointing mechanism since our browser tab was unreliable to continue the session and GCP instances were prone to die time to time. As we switched to larger models, Checkpointing was conducted mid-epoch based on step counts to prevent data loss.

5.4 Results

5.4.1 Sentiment Analysis

Table 1 displays the results for Full Model Fine-Tuning. We observed massive variance based on the learning rate. Table 2 shows results for Last-Linear layer fine-tuning, which is much more robust to higher learning rates.

Model Configuration	Learning Rate	SST Dev (/ Test)	CFIMDB Dev
STAFF-BASELINE	—	0.513	0.976
GPT-2 Base (117M)	1×10^{-3}	0.490	0.829
GPT-2 Base (117M)	1×10^{-5}	0.514 / 0.542	0.975
GPT-2 XL (1.5B) + FlashAttention-2 + LoRA	1×10^{-3}	0.481	0.963
GPT-2 XL (1.5B) + FlashAttention-2 + LoRA + SimCSE	1×10^{-3}	0.485	0.976

Table 1: Accuracy on Sentiment Analysis datasets (Full Fine-Tuning).

Model Configuration	Learning Rate	SST Dev (/ Test)	CFIMDB Dev
STAFF-BASELINE	—	0.462	0.861
GPT-2 Base (117M)	1×10^{-3}	0.461 / 0.470	0.861
GPT-2 XL (1.5B) + FlashAttention-2 + LoRA	1×10^{-3}	0.555	0.967
GPT-2 XL (1.5B) + FlashAttention-2 + LoRA + SimCSE	1×10^{-3}	0.521	0.922

Table 2: Accuracy on Sentiment Analysis datasets (Last-Linear layer Fine-Tuning).

5.4.2 Paraphrase Detection

Table 3 presents the dev accuracy on the Quora Question Pairs dataset. Scaling the model from GPT-2 Base (117M) to GPT-2 XL (1.5B) using LoRA and FlashAttention-2 was found to improve the performance from 0.897 to **0.903**, suggesting that increased model capacity combined with parameter-efficient fine-tuning can benefit semantic similarity tasks. This also improved our test accuracy from 0.855 for our Base model to **0.901**, one of our test submissions.

We also experimented with augmenting the training objective with a SimCSE-style contrastive loss. While this model achieved a competitive dev accuracy of 0.893, the experiment was limited to three training epochs due to compute constraints. For comparison, the GPT-2 XL + LoRA + FlashAttention-2 model required approximately 19 hours to train for 10 epochs on an NVIDIA A100 40GB GPU with a batch size of 16. The SimCSE variant introduced additional computational overhead from the contrastive objective and required over 2.5 hours per epoch on a Blackwell G4 GPU using the configuration `-batch_size 8 -gradient_accumulation_steps 2 -use_simcse -simcse_weight 0.1 -simcse_temp 0.05`. As a result, we limited the SimCSE experiment to three epochs while still observing comparable performance.

Model Configuration	Epochs	Dev	Test
STAFF-BASELINE	—	0.882	—
GPT-2 Base (117M) Initial	10	0.897	0.855
GPT-2 XL (1.5B) + FA2 + LoRA	10	0.903	0.901
GPT-2 XL (1.5B) + FA2 + LoRA + SimCSE [†]	3	0.893	—

Table 3: Accuracy on the Quora Paraphrase Detection Dev Set.

[†] SimCSE experiment was trained for only three epochs due to compute constraints.

5.4.3 Sonnet Generation

Initial experiments exploring Multi-Token Prediction (MTP) variants across model sizes are summarized in Table 4. We compared standard next-token prediction with both single-head and multi-head MTP objectives across multiple prediction horizons across model sizes. We then integrated LoRA and FlashAttention-2 to enable experiments with larger models while maintaining manageable training costs. Results of these experiments are shown in Table 5. Finally, we performed hyperparameter tuning over learning rate and LoRA configurations. The results of these experiments are summarized in Table 6.

Model	NTP (Baseline)	SH-MTP K=2	SH-MTP K=3	MH-MTP K=2	MH-MTP K=3
GPT-2 Large	41.54	42.46	41.74	42.25	41.89
GPT-2 Medium	41.98	41.77	40.70	41.92	41.90
GPT-2	41.66	41.49	40.94	41.18	41.81

Table 4: ChrF scores for Multi-Head Token Prediction (MTP) variants on the sonnet generation task. SH = Single-Head, MH = Multi-Head. Higher is better.

Model	No LoRA / FA2	LoRA + FA2				
	NTP Baseline	NTP Baseline	SH-MTP K=2	SH-MTP K=3	MH-MTP K=2	MH-MTP K=3
GPT-2 Base	41.54	38.16	39.29	39.18	37.36	38.35
GPT-2 Medium	41.98	39.75	39.03	40.06	37.83	38.84
GPT-2 Large	41.66	39.57	39.69	39.36	41.68	41.64
GPT-2 XL	–	41.06	40.39	38.06	40.84	40.55

Table 5: ChrF scores for MTP variants with LoRA and FlashAttention-2 on the sonnet generation task. SH = Single-Head, MH = Multi-Head. Higher is better.

Table 6: Hyperparameter tuning results (LoRA + FlashAttention-2) on the sonnet generation task. ChrF scores reported; higher is better.

Model	LR	LoRA (r, α)	Objective	chrF
<i>GPT-2 Large</i>				
	$1e-5$	(4, 8)	NTP	41.23
	$2e-5$	(8,16)	NTP	41.03
	$2e-5$	(8,16)	SH-MTP K=2	40.35
	$5e-6$	(8,16)	SH-MTP K=2	39.84
	$1e-5$	(16,32)	SH-MTP K=2	38.29
<i>GPT-2 Medium</i>				
	$1e-5$	(4, 8)	NTP	40.60
	$2e-5$	(8,16)	NTP	40.54
	$2e-5$	(8,16)	SH-MTP K=2	38.25
	$5e-6$	(8,16)	SH-MTP K=2	37.54
	$1e-5$	(16,32)	SH-MTP K=2	38.63

6 Analysis

Learning Rate Sensitivity and Fine-Tuning Stability:

Our sentiment analysis experiments highlight the sensitivity of full-model fine-tuning to the learning rate. As shown in Table 1, when using a learning rate of 1×10^{-3} , the GPT-2 Base model achieved a CFIMDB dev accuracy of 0.829, which is comparable to the staff baseline. Reducing the learning rate to 1×10^{-5} significantly improved performance, increasing CFIMDB dev accuracy to 0.975 and slightly improving SST performance as well. One possible explanation is that large learning rates may cause substantial updates to the pretrained transformer weights before the newly initialized classification head has stabilized, which can make optimization more difficult. In contrast, last-linear-layer fine-tuning (shown in Table 2) keeps the transformer backbone frozen and only trains the classification head. This constraint stabilizes the optimization process and allows the model to tolerate larger learning rates such as 1×10^{-3} without degrading performance, yielding SST dev/test accuracies of 0.461/0.470 while maintaining the CFIMDB baseline performance.

How LoRA Enabled Scaling to Larger Models:

Parameter-efficient fine-tuning using LoRA allowed us to scale from GPT-2 Base (117M) to GPT-2 XL (1.5B) while keeping the number of trainable parameters manageable. As shown in Table 3, this scaling resulted in a modest improvement in dev accuracy from 0.897 to 0.903 on the Quora paraphrase detection task. In our configuration, LoRA updates only approximately 0.626% of the model parameters, enabling the larger model to be trained within available GPU memory and compute constraints.

Although the improvement from scaling is relatively modest, the experiment demonstrates that parameter-efficient fine-tuning can make larger language models practical to train in constrained environments. At the same time, the gains are limited, likely because LoRA restricts updates to a small low-rank subspace of the original model weights.

Effect of Contrastive Learning with LoRA:

We also experimented with augmenting the GPT-2 XL + LoRA setup using a SimCSE-style contrastive loss ($\tau = 0.05$, $\lambda = 0.1$). As shown in Table 3, this configuration achieved a dev accuracy of 0.893 on the paraphrase detection task. While slightly lower than the LoRA-only model trained for the full number of epochs, the SimCSE variant was trained for only three epochs due to compute constraints and still achieved comparable performance.

We hypothesize that when most of the model parameters are frozen under LoRA, a standard classification objective may provide limited supervision for shaping sentence representations. The contrastive objective instead encourages embeddings of paraphrase pairs to be closer in representation space while pushing apart unrelated sentences within the batch. This additional training signal may help guide the latent space toward semantic similarity, complementing the cross-entropy objective used for classification.

Effect of MTP on Sonnet Generation:

As shown in Table 4, the effect of Multi-Token Prediction (MTP) varies across model sizes. For GPT-2 Large, all MTP variants outperform the NTP baseline (41.54), with single-head MTP ($K = 2$) achieving the best score of 42.46. This suggests that larger models have sufficient capacity to benefit from the additional supervision of predicting multiple future tokens, which may help capture short-range structural patterns present in sonnets.

For GPT-2 Medium, MTP variants perform slightly below the NTP baseline (41.98), with the closest result coming from MH-MTP $K = 3$ at 41.90. For GPT-2 Base, most MTP configurations perform slightly worse than the baseline (41.66), with the exception of MH-MTP $K = 3$, which reaches 41.81. These results indicate that the additional prediction objective can introduce optimization difficulty for smaller models, whereas larger models appear better able to utilize the richer training signal provided by MTP.

Impact of LoRA and FlashAttention-2 on Sonnet Generation:

Introducing LoRA and FlashAttention-2 (Table 5) consistently lowers ChrF scores relative to the full fine-tuning baseline across base, medium and large models, which is expected given that LoRA constrains the parameter update to a low-rank subspace. However, GPT-2 XL under LoRA achieves a competitive 41.06 on NTP, suggesting that model scale compensates for the reduced parameter expressiveness of LoRA. Training without LoRA for the GPT-2 XL model was beyond our compute ability so we do not have the relevant baseline to compare to. Positive improvements were seen on GPT-2 Large under Multi-Head MTP ($K = 2$: 41.68, $K = 3$: 41.64), which almost came back and recovered full fine-tuning performance despite the issues from LoRA. This suggests that GPT-2 Large was big enough to take advantage of the richer training signal from multi-head prediction even with LoRA, while smaller models were not.

Single-Head vs. Multi-Head MTP under LoRA:

For base and medium models with LoRA, Single-Head MTP generally outperforms Multi-Head MTP (Table 5). For instance, GPT-2 Medium achieves 40.06 with SH-MTP $K = 3$ but only 37.83 with MH-MTP $K = 2$. This shows that the additional projection heads introduced by Multi-Head MTP add parameters and optimisation complexity that conflict with LoRA’s low-rank constraint for smaller models, and in comparison the simpler single-head approach degrades gracefully. We observe a scaling effect where larger models benefit more from Multi-Head MTP while smaller models benefit more from Single-Head MTP.

Hyperparameter Sensitivity:

Table 6 shows that ChrF scores are moderately sensitive to the choice of learning rate and LoRA rank. For GPT-2 Large, NTP with rank 4 and learning rate 1×10^{-5} achieves the best score of 41.23, while increasing the rank to 16 with Single-Head MTP drops performance to 38.29 which is difference of almost 3 ChrF points. A similar pattern holds for GPT-2 Medium, where the best NTP configuration (40.60) substantially outperforms the worst Single-Head MTP configuration (37.54). This suggests that for the sonnet generation task, higher LoRA rank does not reliably help and may even hurt, possibly because the increased rank overfits to the training distribution or destabilises training dynamics. The learning rate 2×10^{-5} with rank 8 performs competitively across both model sizes.

Qualitative Failure Modes:

Also the table 6 tells us that the degradation under $K = 3$ MTP and higher LoRA ranks likely comes

because sonnets was losing structural coherence where model might have produced grammatically plausible lines which failed to maintain rhyme schemes. Future work could investigate this further by evaluating generated sonnets using an explicit rhyme-consistency metric alongside chrF.

7 Conclusion

In this project, we modernized a GPT-2 baseline by scaling model capacity and integrating several recent training and architectural techniques. We showed that parameter-efficient fine-tuning using LoRA, combined with memory-efficient attention through FlashAttention-2, enables scaling to GPT-2 XL (1.5B parameters) within the compute constraints typical of academic environments.

Across tasks, our experiments highlight that model scaling alone does not always lead to substantial improvements. Instead, incorporating additional training objectives and architectural modifications—such as contrastive learning and multi-token prediction—can help guide model representations and generation behavior in ways that complement parameter scaling. These results suggest that combining parameter-efficient scaling with targeted training objectives helps improve both semantic understanding and structured generation in GPT-style models.

Teamwork Contribution

We both worked on the implementation of the base model for the milestone, splitting sub tasks. After the milestone, the extension tasks were split mainly to make better use of the compute credits we had: Manan focused on the Paraphrase Detection and Sentiment Analysis tasks, while Sanjay focused on the Sonnet Generation task.

Throughout the project, we stayed in close contact with a lot of overlap, constantly sharing ideas on what experiments to try, what to avoid, how to set up training, and which machines to use. We both also contributed to writing the report and analysis. The extension task split was to make better use of individual compute credits and to explore our techniques in depth. The Paraphrase Detection task took up a lot more time and GPU resources, so we could not run as many experiments there. The smaller dataset of the Sonnet Generation task and the smaller models gave us more room to try things out, tune hyperparameters, and learn more through experimentation.

References

- [1] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [2] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.
- [3] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.
- [4] DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- [5] 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.