

From-Scratch GPT-2 and Efficient Adaptation

Stanford CS 224N Default Project

Bryan Alexis Pineda
Department of Computer Science
Stanford University
bamp86@stanford.edu

Michael James Nixon
Department of Computer Science
Stanford University
mnixon@stanford.edu

TA mentor: Advit Deepak (advit@stanford.edu)

Abstract

We implement GPT-2 from scratch in PyTorch, including causal multi-head self-attention, Pre-LayerNorm transformer blocks, weight tying, and a custom AdamW optimizer. We validate the implementation against pretrained HuggingFace weights and study adaptation on two tasks: paraphrase detection on Quora Question Pairs and sonnet generation on Shakespeare. Our experiments proceed in stages: dense full fine-tuning baselines, an initial matched comparison of LoRA, DoRA, and ReFT, a LoRA-only ablation over adapter placement and decoding, and a final extended re-benchmark with a stronger training recipe. Full fine-tuning remains the strongest dense baseline, while Extended DoRA gives the best overall PEFT results, reaching 0.883 accuracy on paraphrase detection and 41.683 dev chrF on sonnet generation. Across the study, we find that method choice, adapter placement, training recipe, and decoding policy all materially affect performance.

1 Introduction

Large language models built on the transformer architecture [1] are now the standard foundation for modern NLP, but many of their practical tradeoffs become clear only when the model is implemented and adapted end to end. GPT-2 [2] is a useful setting for that study: it is large enough to expose the core engineering and modeling decisions behind decoder-only language models, yet still manageable for controlled experiments on downstream tasks.

In this project, we implement GPT-2 from scratch in PyTorch. Our model includes causal multi-head self-attention, Pre-LayerNorm transformer blocks with GELU feed-forward networks, learned token and positional embeddings, weight-tied output projections, and a custom AdamW optimizer with bias correction and decoupled weight decay [3, 4]. We verify the implementation against pretrained HuggingFace weights before using it as the base model for a multi-stage adaptation study.

We focus on two downstream tasks with different demands. The first is **paraphrase detection** on Quora Question Pairs [5], where the model must produce a sequence-level semantic judgment from paired question inputs. The second is **sonnet generation**, where the model is fine-tuned on Shakespeare’s sonnets and asked to continue held-out prefixes under autoregressive decoding with nucleus sampling [6], evaluated with chrF [7]. Together, these tasks let us study GPT-2 as both a classifier and a generator.

Our final project is not only about reproducing GPT-2, but about understanding how different adaptation choices change performance. We first establish dense full fine-tuning baselines, then compare three parameter-efficient fine-tuning (PEFT) methods, LoRA [8], DoRA [9], and ReFT [10], under a matched setup. After that initial comparison, we run a focused LoRA ablation to test the effects of adapter placement, optimization, and decoding policy, then re-benchmark all three PEFT methods under an extended setup with stronger pooling, loss design, scheduling, and adapter coverage. This staged design leads to the main conclusion of the report: PEFT can approach dense fine-tuning,

but the outcome depends on more than the choice of PEFT family alone. Adapter placement, training recipe, and decoding strategy all matter, and under our final setup DoRA gives the strongest overall PEFT results.

2 Related Work

GPT-2. Radford et al. [2] introduced GPT-2, a 1.5B parameter decoder-only transformer trained on WebText. Unlike BERT [11], which uses bidirectional attention and is pre-trained with masked language modeling, GPT-2 uses causal (left-to-right) attention and is trained with a standard language modeling objective. This makes GPT-2 naturally suited for generation tasks and adaptable to classification via cloze-style prompting. We implement the base GPT-2 configuration (12 layers, 768 hidden dimensions, 12 attention heads) and initialize from publicly available pretrained weights.

Optimization. The Adam optimizer [12] adapts learning rates using first and second moment estimates of gradients. AdamW [4] decouples weight decay from the gradient update, applying it directly to the parameters rather than incorporating it into the gradient. This leads to more consistent regularization across parameter groups. We implement AdamW from scratch and pair it with a cosine annealing schedule with linear warmup [13], which has been shown to improve convergence in transformer fine-tuning.

Text Generation. Holtzman et al. [6] demonstrate that beam search and greedy decoding lead to degenerate, repetitive text. They propose nucleus (top- p) sampling, which restricts the sampling distribution to the smallest set of tokens whose cumulative probability exceeds p , renormalizing over this nucleus. This produces more diverse and coherent long-form text. We use nucleus sampling with temperature scaling for sonnet generation.

Paraphrase Detection. Prior work on paraphrase detection includes sentence embedding approaches [14] and cross-encoder fine-tuning on BERT-style models. We take a generative approach, framing the task as next-token prediction following a natural language prompt, avoiding the need for a separate classification architecture.

Parameter-Efficient Fine-Tuning. LoRA [8] freezes pretrained weights and learns low-rank adapter matrices, reducing the number of trainable parameters while at the same time preserving downstream performance. DoRA [9] extends LoRA by decomposing the effective weight update into independent magnitude and direction components. ReFT [10] takes a different approach entirely, leaving weights unchanged and instead applying learned low-rank interventions to hidden representations via forward hooks.

3 Approach

3.1 GPT-2 Architecture

Our GPT-2 implementation follows the original specification [2]. The model consists of three main components.

Transformer Blocks. The model stacks L identical transformer blocks, each applying Pre-LayerNorm [3]. Each block consists of causal multi-head self-attention followed by a position-wise feed-forward network with GELU activation, with residual connections and dropout around each.

Causal Self-Attention. We implement multi-head attention with separate query, key, and value projections. We use PyTorch’s `F.scaled_dot_product_attention` with `is_causal=True`, which enforces that each token only attends to prior positions. On PyTorch 2.2+, this automatically uses the FlashAttention kernel.

Output. Lastly, as the last step, a layer normalization is applied. Output logits are computed by projecting hidden states through the transposed word embedding matrix. In the base classification setup, we use the hidden state of the last non-padding token.

3.2 AdamW Optimizer

We implement AdamW [4] from scratch. At each step t :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad (1)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \quad (2)$$

$$\hat{\alpha}_t = \alpha \cdot \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}, \quad (3)$$

$$\theta_t = \theta_{t-1} - \hat{\alpha}_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} - \alpha \cdot \lambda \cdot \theta_{t-1}. \quad (4)$$

The weight decay λ is applied directly to the parameters, and not folded into the gradient. We specifically used $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$.

3.3 Paraphrase Detection

We frame paraphrase detection as binary classification. Each pair of sentences is formatted as a natural language prompt:

Is “{s1}” a paraphrase of “{s2}”? Answer “yes” or “no”:

In our initial full fine-tuning and PEFT experiments, we use a linear classification head on top of GPT-2 representations. In the final extended setup, we replace last-token pooling with mean pooling over all non-padding positions before the classifier, which captures richer sequence-level semantics for longer question-pair inputs. A dropout layer (rate 0.1) is applied before the linear head for regularization.

We also introduce several training improvements in the final extended PEFT re-benchmark. The loss uses label smoothing ($\epsilon = 0.1$) and class-weighted cross-entropy with weights [1.0, 63/37], compensating for the Quora dataset’s class imbalance ($\approx 63\%$ non-paraphrase). The learning rate follows a linear warmup over the first 10% of steps followed by cosine decay to zero, and we apply weight decay $\lambda = 0.01$. Training data is doubled by appending sentence-swapped pairs (s_2, s_1, label) for each (s_1, s_2, label), exploiting the symmetry of the paraphrase relation. The best checkpoint is selected by macro F1 on the development set rather than accuracy, which is more robust to class imbalance.

3.4 Sonnet Generation

We fine-tune GPT-2 on Shakespeare’s 154 sonnets using next-token cross-entropy loss. At inference, given the first three lines of a held-out sonnet as a prompt, we generate completions using nucleus (top- p) sampling [6] with temperature scaling. To reduce repetition on the small training corpus, we apply a repetition penalty of 1.3, logits of tokens already present in the generated prefix are divided (if positive) or multiplied (if negative) by this factor before sampling. Generation terminates at the EOS token or after 14 non-empty lines. We evaluate generated continuations against ground-truth continuations using chrF score [7], computed with sacrebleu.

3.5 Parameter-Efficient Fine-Tuning

We implement three PEFT methods targeting the attention layers of the GPT-2 transformer. In all methods the pretrained GPT-2 weights are frozen, only the adapter parameters are trained.

LoRA. LoRA adds a trainable low-rank perturbation to a frozen weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$:

$$h = W_0 x + \frac{\alpha}{r} B A x,$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$. A is initialized with Kaiming uniform and B is initialized to zero, ensuring the adapter contributes nothing at initialization. In the initial PEFT benchmark, we apply LoRA to the query and value projections of every attention layer with rank $r = 8$ and scaling factor $\alpha = 16$. In the final extended setup, we broaden coverage to the query, key, value, and output-dense projections with rank $r = 16$.

DoRA. DoRA decomposes the effective weight matrix into a learnable per-output magnitude vector $m \in \mathbb{R}^{d_{\text{out}}}$ and a normalized direction matrix:

$$W' = m \odot \frac{W_0 + \Delta W}{\|W_0 + \Delta W\|_c},$$

where $\Delta W = \frac{\alpha}{r}BA$ is the same LoRA update, $\|\cdot\|_c$ denotes per-row ℓ_2 normalization, and m is initialized from the row norms of W_0 . This allows magnitude and direction to be adapted independently. As with LoRA, our initial DoRA benchmark uses Q+V adapters with rank 8, while the final extended setup uses Q+K+V+O adapters with rank 16.

ReFT. Rather than modifying weights, ReFT applies learned interventions to hidden representations via forward hooks registered on each transformer layer. After each hooked layer, the hidden state h at the last non-padding position is updated as:

$$h \leftarrow h + R \tanh(L \text{LayerNorm}(h)),$$

where $L \in \mathbb{R}^{r \times d}$ is initialized with Kaiming uniform and $R \in \mathbb{R}^{d \times r}$ is initialized to zero. Because weights are never modified, ReFT introduces no overhead at weight-loading time.

4 Experiments

4.1 Datasets

Sentiment Classification. We evaluate on two sentiment datasets as a sanity check for our GPT-2 implementation. The Stanford Sentiment Treebank (SST) [15] contains 8,544 training and 1,101 development sentences with 5-class sentiment labels [0,4]. CFIMDB contains 1,707 training and 245 development examples with binary sentiment labels.

Paraphrase Detection. We use the Quora Question Pairs dataset [5], containing 283,003 training and 40,429 development question pairs labeled as paraphrase [1] or non-paraphrase [0].

Sonnet Generation. We fine-tune on 154 of Shakespeare’s sonnets, and evaluated via chrF score [7].

4.2 Experimental Setup

All experiments use the base GPT-2 configuration: $d = 768$, $L = 12$ layers, $H = 12$ attention heads, and AdamW optimization. Gradient norms are clipped to 1.0. The cosine annealing schedule with 10% linear warmup and weight decay $\lambda = 0.01$ are introduced only in the final extended setup in Experiment C.

We organized the study in four stages. First, we fully fine-tuned GPT-2 on paraphrase detection and sonnet generation to establish dense baselines. Next, in Experiment A, we benchmarked three PEFT methods, LoRA, DoRA, and ReFT, under a matched setup to test whether task-specific adaptation with far fewer trainable parameters could remain competitive. In Experiment B, we narrowed to LoRA for scope and ran a deeper study of what matters beyond PEFT family choice, including adapter placement and optimization. Experiment B(ii) isolates decoding effects for sonnet generation. Finally, in Experiment C, we added a set of extensions and re-benchmarked all three PEFT methods: mean pooling over token representations instead of last-token pooling for paraphrase, class-weighted cross-entropy with label smoothing ($\epsilon = 0.1$), sentence-swap data augmentation, cosine learning-rate scheduling with 10% warmup, weight decay ($\lambda = 0.01$), and expanded LoRA/DoRA adapter coverage to Q, K, V, and output-dense projections at rank 16.

4.3 Sentiment Classification

Table 1 reports results for three fine-tuning modes. The *last-linear-layer* mode freezes all GPT-2 parameters and only trains a linear classification head. The *full-model* mode fine-tunes all parameters. The *LoRA* mode freezes GPT-2 weights and trains only low-rank adapters on Q and V projections.

Table 1: Sentiment classification dev accuracy.

Fine-tuning Mode	SST Dev Acc	CFIMDB Dev Acc
Last Linear Layer	0.459	0.869
Full Model	0.525	0.976
LoRA ($r=8$)	0.385	0.555

4.4 Paraphrase Detection

Table 2 reports the full progression for paraphrase detection: the full fine-tuning baseline, the matched PEFT comparison from Experiment A, the LoRA-only follow-up from Experiment B, and the final re-benchmark from Experiment C. The dense full fine-tune remains our best overall dev result at 0.894. In Experiment A, LoRA and DoRA are close and both substantially outperform ReFT. Experiment B then shows that LoRA can be improved by adapting more projections: Q+K+V adapters on all layers reach 0.870 mean dev accuracy across two seeds. The final extended setup changes the PEFT ranking again: after adding the revised pooling and training recipe and expanding adapter coverage, Extended DoRA becomes our best PEFT paraphrase model at 0.883 dev accuracy and 0.883 test accuracy, with Extended LoRA close behind at 0.882 dev accuracy.

Table 2: Paraphrase detection accuracy on Quora dev set.

Model	Setting	Dev Acc	Test Acc
Staff Baseline	Reference	0.882	0.856
<i>Our Models (Initial Setup)</i>			
Full fine-tune		0.894	0.840
LoRA	Q+V, all layers, $r = 8$	0.866	
DoRA	Q+V, all layers, $r = 8$	0.863	
ReFT	rank 8 intervention	0.817	
Best LoRA sweep	Q+K+V, all layers, $r = 8$ (mean over 2 seeds)	0.870	
<i>Our Models (Extended Setup)</i>			
Extended LoRA	Q+K+V+O, all layers, $r = 16$	0.882	
Extended DoRA	Q+K+V+O, all layers, $r = 16$	0.883	0.883
Extended ReFT	rank 8 intervention	0.797	

4.5 Sonnet Generation

Table 3 reports sonnet generation chrF on the dev set. As with paraphrase detection, the table is organized by experimental stage. The full fine-tune provides the dense baseline. In Experiment A, the three PEFT methods are tightly clustered around 37.2–37.5 chrF, which motivated the deeper LoRA-only study. Within Experiment B, the best LoRA structure is smaller than the initial all-layer baseline: adapting only Q+V in the top 6 layers raises mean dev chrF to 37.719, and tuning learning rate and dropout on top of that structure reaches 39.958. Experiment C changes the picture again. After re-benchmarking all three PEFT methods with the extended setup, Extended DoRA becomes the strongest PEFT sonnet model at 41.683 dev chrF and 40.053 test chrF, while the full fine-tune still gives our best reported test chrF at 40.583.

Decode sweep. Experiment B(ii) isolates inference-time effects by studying decoding for the sonnet task. For each of eight LoRA source runs, we took the top 4 checkpoints and decoded them under four fixed policies: three stochastic sampling settings and one greedy setting. Table 4 reports mean dev chrF for each policy. Performance improves steadily as sampling is made less restrictive: temperature 0.7 with top- $p = 0.90$ and top- $k = 40$ reaches 34.71 chrF, temperature 0.8 with top- $p = 0.95$ and top- $k = 50$ reaches 37.01, and temperature 0.9 with top- $p = 0.95$ and no top- k truncation performs best at 38.52. Greedy decoding is much worse at 28.84 chrF, showing that sampling policy is a major determinant of generation quality on this task.

Table 3: Sonnet generation chrF scores on held-out dev and test sets.

Model	Setting	Dev chrF	Test chrF
Staff Baseline	Reference	41.414	41.208
<i>Our Models (Initial Setup)</i>			
Full Fine-Tune		41.541	40.583
LoRA	Q+V projections, all layers, $r = 8$	37.249	
DoRA	Q+V projections, all layers, $r = 8$	37.530	
ReFT	Rank-8 intervention	37.204	
Best LoRA structure sweep	Q+V projections, top 6 layers, $r = 8$	37.719	
Best LoRA optimization sweep	Q+V projections, top 6 layers, $r = 8$, lr 3×10^{-4} , dropout 0.05	39.958	
<i>Our Models (Extended Setup)</i>			
Extended LoRA	Q,K,V,O projections, all layers, $r = 16$	39.381	
Extended DoRA	Q,K,V,O projections, all layers, $r = 16$	41.683	40.053
Extended ReFT	Rank-8 intervention	39.944	

Notes. Unless otherwise specified, results use random seed 11712. Structure and optimization sweeps report the mean over two seeds.

Table 4: Sonnet decode sweep results.

Policy	Temp.	Top- p	Top- k	Mean Dev chrF
Sampling	0.9	0.95	0	38.524
Sampling	0.8	0.95	50	37.006
Sampling	0.7	0.90	40	34.708
Greedy (no sampling)	1.0	1.00	0	28.844

5 Analysis

The staged design of the experiments matters for interpretation. The full fine-tuning runs establish that dense adaptation remains a strong baseline for both tasks. Experiment A then asks whether PEFT can recover similar performance with far fewer trainable parameters. Under that matched initial setup, LoRA and DoRA are close on both tasks, while ReFT is consistently weaker on paraphrase detection and only roughly tied on sonnet generation. That initial result made a LoRA-only follow-up reasonable as a scoped study, but it should be read as a scoping decision rather than as a final verdict that LoRA is globally best.

Experiment B shows that factors beyond PEFT family choice have large effects. For paraphrase detection, LoRA improves from 0.866 to 0.870 mean dev accuracy once adapter coverage is broadened to Q+K+V across all layers, suggesting that the classification task benefits from additional adapter capacity. For sonnet generation, the opposite pattern appears within the LoRA-only sweep: a lighter Q+V adapter restricted to the top 6 layers outperforms the all-layer LoRA baseline, and modest optimization changes then raise mean dev chrF from 37.719 to 39.958. So within LoRA, the two tasks prefer different adapter structures.

The decode sweep makes the inference story even sharper. Holding the source LoRA runs fixed, changing only the sampling policy moves mean dev chrF from 28.844 with greedy decoding to 38.524 with temperature-0.9, top- $p = 0.95$ sampling. That nearly 10-point spread is larger than many of the training-time gains in the LoRA sweep, so generation quality cannot be understood from checkpoint loss or greedy decoding alone.

Finally, Experiment C updates the overall conclusion. Once we add mean pooling, class-weighted label-smoothed loss, sentence-swap augmentation, cosine scheduling with warmup, weight decay, and broader Q+K+V+O rank-16 adapters, DoRA becomes the strongest PEFT method in the final report on both tasks: 0.883 dev/test accuracy on paraphrase detection and 41.683 dev chrF on sonnet generation. Extended LoRA remains competitive, especially on paraphrase detection, but it no longer

defines the final best PEFT result. The evidence therefore supports a more nuanced takeaway than our mid-project narrative did: early on, adapter placement and decoding looked more important than PEFT family choice, but after improving the training recipe and re-running all three methods, method choice matters as well.

6 Conclusion

We built GPT-2 from scratch in PyTorch, validated it against pretrained weights, and applied it to classification and generation tasks. Our experiments proceeded in stages: dense full fine-tuning baselines, an initial matched PEFT comparison, a scoped LoRA-only ablation with a decode sweep, and a final extended PEFT re-benchmark. That sequence matters. The initial comparison justified the LoRA-only follow-up for scope because LoRA and DoRA were close under the matched setup, but the final extended results do not support the stronger claim that LoRA is the best PEFT method overall.

Our final results show that the strongest PEFT results come from Extended DoRA: 0.883 dev/test accuracy on paraphrase detection and 41.683 dev chrF with 40.053 test chrF on sonnet generation. Full fine-tuning still provides our strongest dense baseline, including the best reported sonnet test chrF of 40.583. The LoRA-only experiments remain useful because they show that adapter placement and optimization materially affect performance, and the decode sweep shows that sampling policy can change sonnet chrF by nearly 10 points even with the source checkpoints held fixed. Overall, PEFT can approach dense fine-tuning with far fewer trainable parameters, but the final outcome depends jointly on method choice, adapter placement, training recipe, and decoding policy.

Team contributions

Both authors jointly designed the project, implemented the core GPT-2 model and optimizer, ran the downstream experiments, analyzed the results, and wrote the report. The project infrastructure, PEFT experiments, and final write-up were completed collaboratively.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- [2] 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.
- [3] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. *International Conference on Machine Learning*, 2020.
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.
- [5] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question pairs. <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>, 2017.
- [6] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *International Conference on Learning Representations*, 2020.
- [7] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

- [8] 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. *International Conference on Learning Representations*, 2022.
- [9] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. *International Conference on Machine Learning*, 2024.
- [10] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. ReFT: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 2024.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, 2019.
- [12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.
- [13] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations*, 2017.
- [14] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of EMNLP*, 2019.
- [15] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.