Improving GPT-2 Task-Specific Performance via Impossible Distillation, Sparse Attention, and Parameter-Efficient Fine-Tuning

Stanford CS224N Default Project

David F. Castro-Pena
dcastrop@stanford.edu

Muyin Yao
yaomuyin@stanford.edu

Sylvia Sun
ys3835@stanford.edu

Abstract

In this project, we reproduced the Generative Pre-trained Transformer 2 (GPT-2) model and fine-tuned it for various downstream tasks, introducing task-specific improvements to both the model architecture and fine-tuning techniques. We implemented GPT-2 from scratch, loaded pretrained weights from Hugging Face, and fine-tuned the model for sentiment analysis, paraphrase detection, and sonnet generation.

- Paraphrase Detection: We explored Impossible Distillation, a novel knowledge distillation framework that enables high-quality paraphrase detection using a weak teacher model. Unlike conventional distillation approaches that rely on large-scale models like GPT-3, Impossible Distillation [1] leverages GPT-2's paraphrastic proximity property to iteratively generate high-quality paraphrase pairs through self-distillation. Our results show that models trained with Impossible Distillation outperform both standard GPT-2 fine-tuning and ChatGPT-based distillation on paraphrase detection tasks.
- Sonnet Generation: We evaluated various decoding strategies, including greedy search, top-k sampling, and beam search. Furthermore, we explored parameter-efficient fine-tuning (PEFT) techniques, specifically LoRA [2], to enhance generation quality while minimizing computational overhead. Alongside these methods, we also investigated integrating *Sparse Attention* into the generative pretraining using Poetry Foundation Corpus [3] for improved performances in sonnet generation. Our results show that additional pretraining on a poetry-rich dataset before fine-tuning could help the model internalize rhyme, meter, and poetic constraints more effectively.

We selected the best modifications from each category and combined them into generating the final results, achieving a test performance of 0.872 on paraphrase detection and 41.822 CHRF score on sonnet generation.

1 Introduction

GPT-2 has demonstrated strong performance across various NLP tasks, but its high computational cost and fine-tuning inefficiencies pose challenges for broader applications. Its self-attention mechanism scales quadratically with sequence length, making training and inference computationally expensive. Additionally, traditional fine-tuning requires updating all model parameters, which increases the risk of overfitting and inefficiency. Meanwhile, knowledge distillation typically relies on large teacher models like GPT-3, limiting its accessibility.

In this project, we explored methods to improve GPT-2's efficiency and task-specific performance in paraphrase detection and sonnet generation. We implemented Impossible Distillation, a self-distillation framework that enhances paraphrase detection by leveraging paraphrastic proximity. To reduce computational overhead, we integrated Sparse Attention, which selectively attends to a subset of tokens, improving efficiency in attention computation. We also evaluated various decoding strategies to optimize text fluency and coherence. Additionally, we investigated LoRA-based fine-tuning, a parameter-efficient adaptation technique that updates only a small fraction of model

parameters, significantly reducing resource requirements. To further enhance sonnet generation, we explored pretraining on poetry-rich datasets.

By combining self-distillation, attention optimizations, and efficient fine-tuning, this project demonstrates that GPT-2 can achieve strong performance while maintaining reasonable computational costs. Our findings provide insights into making large language models more scalable, adaptable, and resource-efficient for diverse NLP applications.

2 Related Work

Knowledge Distillation in NLP. Knowledge distillation (KD) transfers knowledge from large teacher models, such as GPT-3, to smaller student models. Common KD methods train students to mimic the softened probability distributions of powerful teachers. Techniques like self-distillation and multi-stage distillation enhance generalization and reduce computational costs.

Jung et al. (2024) introduced Impossible Distillation, leveraging paraphrastic proximity in smaller models like GPT-2. This method exploits natural paraphrase proximity within a model's distribution without requiring explicitly trained large-scale teachers. Impossible Distillation employs iterative self-refinement: generating paraphrase candidates, filtering via semantic and diversity criteria, fine-tuning students on these paraphrases, and refining through self-distillation. This approach yields superior results compared to models distilled from larger models like ChatGPT, reducing computational demands significantly, and benefiting resource-limited NLP tasks such as paraphrase detection and summarization.

Prior paraphrase detection research largely employed transformer-based models (BERT, RoBERTa, GPT-2), typically fine-tuned on datasets like Quora Question Pairs (QQP). While traditional methods train classifiers on sentence embeddings, recent work explores generative paraphrase evaluation. Methods like Impossible Distillation provide hybrid solutions, blending generative efficiency with classification robustness.

Sparse Attention. Sparse attention techniques mitigate the quadratic computational complexity of traditional transformer self-attention. Child et al. [4] proposed the Sparse Transformer, using fixed sparse attention patterns to lower computational and memory costs. Beltagy et al. [5] extended this by developing Longformer, which combines local windowed attention with select global attention tokens, efficiently capturing long-range document dependencies.

Decoding Strategies. For text generation tasks, decoding is the process of selecting the next word based on a probabilistic distribution over the vocabulary. Various decoding strategies have been proposed to achieve good accuracy and diversity in generated text. Traditional approaches like greedy decoding and beam search prioritize high-probability sequences but can lead to repetitive or deterministic outputs. To introduce variability, sampling-based methods such as top-k sampling and nucleus sampling have been explored. These methods limit selection to the most probable yet dynamic subset of tokens. More recent work has investigated dynamically adjusted decoding strategies or reinforcement learning-based approaches to further enhance fluency and relevance. Decoding strategies play a critical role in optimizing generation quality.

Efficient Adaptations. Fine-tuning large pretrained models presents two main challenges. First, full fine-tuning is computationally expensive, requiring significant resources and time. Second, updating all model parameters for a specific downstream task can lead to overfitting, potentially hurting the generalization ability gained during pretraining.

To address these issues, various efficient adaptation methods have been proposed. Techniques such as layer freezing preserve the general knowledge of pretrained representations by restricting updates to only a subset of layers. Adapter layers introduce lightweight task-specific modules within the model while keeping the core parameters fixed. Additionally, LoRA [2] and other techniques such as prefix tuning introduce a minimal number of trainable parameters, enabling efficient adaptation while maintaining performance. These approaches significantly reduce computational overhead and has proven to even improve generalization.

3 Approach

Our goal is to fine-tune GPT-2 for multiple downstream tasks while optimizing both computational efficiency and performance. We fine-tune GPT-2 separately for paraphrase detection and sonnet generation.

For paraphrase detection, we incorporate LogSparse Attention [6] to mitigate the quadratic complexity of self-attention, improving efficiency while maintaining accuracy. Additionally, we implement Impossible Distillation, a self-distillation framework that iteratively refines paraphrase quality without relying on a large teacher model.

For sonnet generation, we evaluate various decoding strategies (e.g., greedy search, top-k sampling, beam search) to assess their impact on CHRF scores. We also explore LoRA-based fine-tuning to balance model performance and computational cost when fine-tuning larger GPT-2 variants. To further enhance the model's stylistic and structural understanding of poetry, we incorporate generative pretraining on a poetry-rich dataset before fine-tuning on Shakespearean sonnets.

3.1 Baselines

Our baseline GPT-2 model loads pretrained weights from Hugging Face [7] and is fine-tuned separately for each task. Each task is handled via a separate fine-tuning pipeline that adapts the pre-trained weights using task-specific datasets. For paraphrase detection, we fine-tune GPT-2 on the Quora Question Pairs dataset, where the model learns to classify if two sentences are paraphrases. For sonnet generation, we fine-tune GPT-2 on the Shakespearean Sonnets dataset, where the model learns to finish a sonnet. No additional modification is done to both fine-tuning pipelines.

3.2 Impossible Distillation for Paraphrase Detection

The Impossible Distillation framework, introduced by Jung et al. (2024), provides an alternative approach to knowledge distillation that does not rely on a large teacher model. Instead, it leverages the paraphrastic proximity property of smaller pre-trained models such as GPT-2, enabling effective paraphrase generation and classification without requiring a large-scale teacher model.

To formalize the self-refinement process, let x be an input sentence and y^* be the ideal paraphrase. The student model S_θ generates a paraphrase candidate y':

$$y' \sim P_\theta(y|x),$$

where $P_\theta(y|x)$ is the probability distribution of the student model.

A critic model C_ϕ evaluates the quality of y' by computing a semantic similarity score:

$$s = C_\phi(x, y').$$

Only paraphrases with scores above a threshold τ are retained for training:

$$\mathcal{D}_{filtered} = \{(x, y')|s > \tau\}.$$

The student model is then fine-tuned on the filtered dataset using a combination of cross-entropy loss $\mathcal{L}_{CE}$ and a knowledge distillation loss $\mathcal{L}_{KL}$:

$$\mathcal{L} = \mathbb{E}_{(x,y')\sim\mathcal{D}_{filtered}} \left[\mathcal{L}_{CE}(S_\theta(y'|x)) + \lambda\mathcal{L}_{KL}(S_\theta||P_T)\right],$$

where P_T is the initial probability distribution from the pretrained GPT-2 model, and λ controls the influence of the distillation loss.

This iterative self-distillation process continues, gradually improving the model's ability to generate high-quality paraphrases. By leveraging internal model representations and a structured refinement process, Impossible Distillation enables efficient paraphrase detection with fewer computational resources while maintaining high accuracy.

3.3 Sparse Attention

Given a query Q, key K, value V, and d_k is the dimensionality of the key vectors, for each token i, we define a subset of tokens $S(i)$ that token i is allowed to attend to. The attention for token i is then computed as:

$$\text{Attention}(Q_i, K_{S(i)}, V_{S(i)}) = \text{softmax}\left(\frac{Q_i K_{S(i)}^\top}{\sqrt{d_k}}\right) V_{S(i)},$$

To determine the subset of tokens, we referenced the LogSparse Attention [6]. The subset of past tokens is defined by an exponential (logarithmic) spacing. For each token at position l, define the set of indices

$$I(l) = \{\, l - 2^{\lfloor \log_2(l) \rfloor},\ l - 2^{\lfloor \log_2(l) \rfloor - 1},\ \ldots,\ l - 2^0,\ l \,\},$$

where $\lfloor \cdot \rfloor$ denotes flooring. Then, the attention for token l is computed as Attention$(Q_l, K_{I(l)}, V_{I(l)})$ where $K_{I(l)}$ and $V_{I(l)}$ are the key and value vectors corresponding only to the indices in $I(l)$. Since $|I(l)| = O(\log l)$, the per-token memory cost is reduced to $O(\log L)$ and the overall cost to $O(L \log L)$. Note, to ensure all token can eventually propagate to any other token, enough $(O(\log L))$ layers needs to be stacked. Currently, we kept the 12 stacked transformer layers structure, and plan to explore increasing the density of sparse connection (i.e start sparse attention k steps away from the current position) or use a hybrid approach (i.e interleaving sparse and dense attention layers). The sparse attention mechanism was integrated into the existing attention module.

3.4 Generative Pre-training

For pretraining, we use the Poetry Foundation Poems dataset to enhance the model's linguistic and stylistic understanding. We created a custom dataloader tailored for the Poetry Foundation Poems dataset. The training objective here is to predict the next token given a sequence, which enables the model to internalize the nuances of poetic language.

3.5 Decoding Strategies

To improve generation quality, we explored below decoding strategies:

1. Greedy: Select the token with the highest probability at each step.
2. Top-P: Randomly select from a set of candidates with the highest probability and the sum of their probability exceed threshold P.
3. Top-K: Randomly select from a set of K candidates with the highest probability.
4. Beam-search: Probe N steps into the future. At each decoding step, select top H candidates based on cumulative probability. Select the candidate with the highest probability sequence.
5. Top-K-P: Randomly select from a set of K candidates with the highest probability and the sum of their probability exceed threshold P.

3.6 Efficient Adaptations

To leverage larger GPT2 models efficiently, we explored LoRA (Low-Rank Adaptation), a parameter-efficient fine-tuning method that injects trainable low-rank matrices into pre-existing weight layers while keeping the original model parameters frozen.

We updated our GPT2 implementation to add necessary interfaces, and applied hugging-face PEFT (Parameter-Efficient Fine-Tuning) library [8] to transform our GPT2 into one with LoRA. To demonstrate the effect of LoRA, we focus on GPT-2 Large model with 812M parameters, LoRA is applied to all 12 attention layers, and explored the impact of different rank.

4 Experiments

4.1 Data

We will use the datasets provided in the project handout which includes the Stanford Sentiment Treebank (SST) dataset, CFIMDB dataset, the Quora dataset, and the Shakespearian sonnets dataset.

- The Stanford Sentiment Treebank (SST) dataset includes 11,855 single sentences from movie reviews, parsed into 215,154 unique phrases, with sentiment labels ranging from negative to positive. It has splits for training (8,544 examples), development (1,101 examples), and testing (2,210 examples).
- The CFIMDB dataset contains 2,434 highly polar movie reviews with binary sentiment labels and is split into training (1,701 examples), development (245 examples), and testing (488 examples).
- Quora dataset consists of question pairs with labels indicating whether particular instances are paraphrases of one another. It has splits with 141,506 training pairs, 20,215 development pairs, and 40,431 test pairs.
- The Shakespearean sonnets contains 156 pieces each consists of 14 lines of text following the rhyme scheme, and is divided into 131 for training, 12 for development, and 13 for testing.

In addition, we use the Poetry Foundation Poems [3] dataset from the Kaggle Community, which contains 13,754 poems extracted from the Poetry Foundation, for additional pretraining before fine-tuning on the sonnet generation task.

4.2 Evaluation method

We evaluate our model using accuracy for sentiment classification and paraphrase detection tasks. For sonnet generation, we use cross-entropy loss on the training sonnets and the chrF metric, a character-level n-gram comparison similar to BLEU that quantifies how closely the output resembles Shakespeare's originals, during evaluation.

4.3 Experimental details

The model configurations of the GPT-2 is explained in the main approach section. Table 1 summarizes key experimental configurations for the three tasks.

Task	Dataset	Learning Rate	Batch Size	Epochs
Sentiment Classification	SST	1×10^{-5}	64	10
	CFIMDB	1×10^{-5}	8	10
Paraphrase Detection	Quora	1×10^{-5}	16	10
Paraphrase Detection: Impossible Distillation	Approach 1 (Quora)	1.5×10^{-5}	8	4
	Approach 2 (Quora)	1.5×10^{-5}	16	8
Sonnet Generation	Shakespearean Sonnets	1×10^{-5}	8	50
Sonnet Generation: Sparse Attention	Shakespearean Sonnets	1×10^{-5}	8	50
Sonnet Generation: Pretraining	Poetry Foundation Poems	1×10^{-5}	4	10
	Shakespearean Sonnets	1×10^{-5}	8	50

Table 1: Summary of experimental configurations for various NLP tasks

The experimental configurations table presents hyperparameter settings across four NLP tasks. Sentiment Classification (SST and CFIMDB datasets), Paraphrase Detection (Quora), and Sonnet Generation (Shakespearean Sonnets) all used a 1×10^{-5} learning rate and 10 epochs, with batch sizes varying (64/8, 16, and 8 respectively). Impossible Distillation experiments on Quora data employed a higher learning rate (1.5×10^{-5}) with two approaches: Approach 1 using smaller batches (8) for fewer epochs (4) and Approach 2 using larger batches (16) for more epochs (8), indicating a methodical exploration of training efficiency trade-offs. For the sonnet generation task, we used the 'gpt' model in all configurations—whether employing sparse attention alone, pretraining alone, or a combination of both. Specifically, we ran 10 epochs of pretraining at a batch size of 4, followed by 50 epochs of fine-tuning with a batch size of 8.

4.4 Results

For sentiment analysis task, the baseline accuracy for both SST and CFIMDB datasets are shown in Table 2. Based on the result, the full model improves over the last-linear-only approach, which is expected. However, SST accuracy seems somewhat low compared to state-of-the-art models (80%+), indicating potential room for further improvement.

4.4.1 Impossible Distillation Results

Our Impossible Distillation experiments yielded promising outcomes. With our modified codebase (Approach 1), we achieved an accuracy of 0.83 on the Quora dataset by implementing refined tokenization, enhanced preprocessing techniques, and contrastive loss to improve paraphrase detection, while training for just 4 epochs with a batch size of 8. In an attempt to further improve results, we developed a second implementation (Approach 2) that extended the training duration to 8 epochs and increased the batch size to 16, incorporating additional filtering mechanisms in the self-distillation process. Interestingly, this approach resulted in a slight decrease in accuracy to 0.82, suggesting that longer training and larger batch sizes may lead to overfitting on this particular task (see Figure 2 and Table 3).

4.4.2 Sparse Attention Results

We also evaluated the effect of applying sparse attention on the paraphrase detection task. The baseline model achieved an accuracy of 0.897, while the model with sparse attention attained a lower accuracy of 0.856. Figure 1 illustrates the training loss and accuracy over epochs for both models. It appears that sparse attention is not performing as well, potentially due to its inability to capture fine-grained local dependencies effectively. That is, if phrase reordering or subtle word substitutions fall outside the log-sparse attention pattern, the model might struggle with semantic equivalence detection. This leads to the direction of integrating a hybrid sparse attention in future works.

4.4.3 Generative Pre-training Results

In addition to implementing sparse attention, we also performed more extensive generative pretraining using the Poetry Foundation Poems dataset. The primary goal of this pretraining step was to enhance the model's understanding of poetic structure, language style, and thematic nuance prior to the Shakespeare sonnet generation phase where previous pre-training step with OpenWebText might be limited. Table 4 in the Appendix summarizes the results using 'gpt' model parameter set. Interestingly, LogSparse Attention did not show the same improvement when combined with pretraining, dropping slightly from 40.483 to 39.901. This is not entirely expected. One possible explanation is that the sparse connectivity pattern in LogSparse may not fully leverage the additional contextual information learned during pretraining—perhaps due to reduced coverage of local dependencies. However, the boost due to adding the additional pretraining is expected.

4.4.4 Decoding Strategies Results

For sonnet generation, we explored greedy decoding, top-p sampling, top-k sampling, a combination of top-p and top-k sampling (top-p-k), and beam search. We evaluated the model's inference ability using a held-out development set of sonnets.

All hyperparameters were set to their default values: we used the 'gpt' model, the number of epochs was set to 10, p was set to 0.9, k to 50, and the temperature to 1.2. For beam search, the number of beams was set to 5, and the probe length was 20.

Table 5 shows the CHRF scores for different decoding strategies. Top-p sampling achieved the highest CHRF score of 41.430. Beam search achieved the lowest score of 26.778.

4.4.5 Adaptation methods

For sonnet generation, we explored various adaptation methods, including freezing some layers and LoRA. For LoRA, we explored the impact of different rank.

All hyperparameters were set to their default values: we used the 'gpt-large' model, the number of epochs was set to 10.

Table 6 shows the number of trainable parameters and average GPU memory usage across various parameter-efficient adaptation configurations. Table 7 presents the performance of different adaptation methods, including final training loss, perplexity, and CHRF scores.

- Freezing the first half of the model reduces the number of trainable parameters and GPU memory usage. LoRA achieves an even greater reduction in trainable parameters while keeping GPU memory usage lower.
- LoRA (r=4, r=8, and r=16) all maintain similar GPU memory consumption (3.34–3.36 GB). However, LoRA (r=32) significantly increases memory usage to 7.55 GB.
- Full fine-tuning achieves the lowest loss (1.556). Freezing the first half of the layers results in lower perplexity (34.215) compared to all LoRA configurations. LoRA (r=16, r=32) achieves CHRF scores of 40.516 and 40.546, respectively, slightly outperforming freezing layers (40.434).

5 Analysis

5.1 Impossible Distillation Results

The Impossible Distillation experiments reveal compelling training dynamics through the detailed visualization in Figure 2. Approach 1 demonstrates remarkably rapid initial learning, as evidenced by its steep accuracy curve that achieves its peak of 0.83 by epoch 4, accompanied by a dramatic decrease in loss from 0.54 to 0.33—a 39 percentage reduction. This efficiency contrasts with Approach 2,

which despite extending training to 8 epochs and employing a larger batch size of 16 (versus 6 for Approach 1), exhibits a noticeably more gradual learning trajectory. The divergent slopes of both accuracy curves during epochs 1-3 particularly highlight this difference in learning velocity. While Approach 2 continues to show incremental improvements beyond epoch 4, it ultimately plateaus at 0.82 accuracy, never quite matching Approach 1's performance despite consuming twice the computational resources. The intersections between the loss and accuracy curves further demonstrate how quickly Approach 1 optimizes, suggesting its parameter configuration may be better suited for this particular task on the Quora dataset. In conclusion, these Impossible Distillation results challenge conventional wisdom about batch size scaling, indicating that smaller batches may sometimes enable faster convergence and higher peak performance, particularly for paraphrase detection tasks. This finding has important implications for resource allocation in model training, suggesting that efficiency gains might be achieved through careful hyperparameter tuning rather than simply extending training duration or increasing computational resources.

5.2 Sparse Attention

Based on our experiments, while sparse attention alone performed comparably to standard self-attention in sonnet generation task, combining sparse attention with additional generative pretraining did not yield the expected improvement. We had hoped that the richer context learned during pretraining would enhance the model's ability to capture long-range dependencies and subtle local nuances; however, the sparse attention mechanism appears to limit the effective capture of fine-grained local dependencies. This limitation may hinder the model from fully leveraging the enhanced contextual representations provided by pretraining, particularly when it comes to tasks that require detailed semantic and syntactic understanding in generating coherent poetic language. These findings suggest that while sparse attention offers computational efficiency, it may need to be integrated with more flexible mechanisms—perhaps through a hybrid attention approach—to fully benefit from the enriched representations obtained during pretraining.

5.3 Generative Pre-training

Generative pretraining alone appears to provide a measurable boost in sonnet generation quality. In our experiments, the baseline self-attention model achieved a CHRF score of 40.517 on the development set, while adding the generative pretraining step raised this score to 41.405. This improvement suggests that pretraining on the Poetry Foundation Poems dataset helps the model internalize richer representations of poetic structure and stylistic nuance, which translates to more coherent and rhythmically consistent sonnet outputs. The additional pretraining likely provides the model with a more refined understanding of language patterns specific to poetry, thus improving its overall performance on the sonnet generation task.

5.4 Decoding Strategies

We initially expected beam search to achieve the best performance. However, it failed to surpass the baseline greedy decoding method. We observed that both greedy decoding and beam search frequently generated repetitive word sequences such as "love love love" or "and and and." This repetition is understandable, as sonnets often contain emotional expressions and filler words. See Table 5 in Appendix. Since sonnet generation requires diversity and creativity, top-p sampling performed the best under these parameter settings.

5.5 Adaptation methods

We compared various methods with the baseline model where we fine-tune the whole mode. First, full fine-tuning remains the best option for performance but is the most resource-intensive and time consuming. For a more memory-efficient approach, LoRA (r=16) provides the best trade-off between efficiency and output quality. Freezing layers, despite reducing GPU usage, leads to performance degradation. Lastly, increasing LoRA rank beyond r=16 does not yield significant improvements, this could be that GPT2 is not large enough and rank=16 is enough to capture the key low-rank adaptations needed for fine-tuning.

Since LoRA constrains weight updates to a low-rank subspace, increasing the rank beyond 16 may not add much expressivity. Another possibility is that GPT2 itself is the limitation. Even if higher-rank LoRA adaptations are applied, the underlying model might not have enough representational power to capture information in higher-rank dimensions. Overall, LoRA (r=16) is the best LoRA configuration, striking the optimal balance between memory efficiency (3.36 GB) and performance (CHRF score).

6 Conclusion

In this project, we explored methods to enhance GPT-2's efficiency and task-specific performance across paraphrase detection and sonnet generation. Our approach incorporated Impossible Distillation for knowledge transfer, Sparse Attention to reduce computational complexity, Pretraining to enhance stylistic adaptation, Decoding Strategies to improve text coherence, and LoRA-based fine-tuning for parameter-efficient adaptation.

For paraphrase detection, Impossible Distillation improved model performance by leveraging self-generated paraphrases without relying on an external teacher. However, we observed that increasing training epochs and batch sizes did not always lead to better results, suggesting the need for careful filtering and selection of paraphrase candidates. Sparse Attention, while computationally efficient, showed limitations in capturing fine-grained semantic relationships, pointing to the need for hybrid sparse-dense attention mechanisms.

In sonnet generation, pretraining on poetry datasets improved the model's ability to capture rhyme, meter, and thematic consistency. Among decoding strategies, top-p sampling provided the best balance between creativity and coherence, while beam search led to excessive repetition. Additionally, LoRA-based fine-tuning proved to be an effective alternative to full fine-tuning, significantly reducing memory usage while maintaining performance.

Further research:

- Multi-task training: While our experiments focused on task-specific fine-tuning, an important direction for future work is multi-task learning. This would result in a more generalizable and powerful model.
- Hybrid attention: Combining sparse and dense attention for more effective semantic modeling.
- Optimized Self-Distillation: Enhancing Impossible Distillation with better paraphrase filtering and adaptive learning strategies.
- Expanded decoding strategy exploration: Wile our experiment only covered a few key decoding strategies, we believe there are values in optimizing creativity by using adaptive decoding methods or reinforcement-learning based decoding strategies.
- Layer-wise adaptive LoRA: Dynamically selecting which layers to apply LoRA to, optimizing efficiency vs. expressiveness.

Our findings suggest several promising directions for further research, to enhance the adaptability, efficiency, and performance of GPT-2 across diverse NLP tasks.

Team Contributions

David F. Castro-Pena: Implemented baseline GPT-2 and Impossible Distillation approach for paraphrasing.
Muyin Yao: Implemented the baseline paraphrase detection and sonnect generation pipelines, explored decoding strategies and incorporated LoRA for the sonnet generation task.
Sylvia Sun: Implemented Sparse attention and incorporated additional generative pretraining for the sonnet generation task.

References

[1] Jaehun Jung, Peter West, Liwei Jiang, Faeze Brahman, Ximing Lu, Jillian Fisher, Taylor Sorensen, and Yejin Choi. Impossible distillation: From low-quality models to high-quality data. *arXiv preprint arXiv:2305.16635*, 2023.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models . `https://arxiv.org/abs/2106.09685`.

[3] Kaggle community. Poetry Foundation Poems. `https://www.kaggle.com/datasets/tgdivy/poetry-foundation-poems/data`.

[4] Rewon Child, Scott Gray, Alec Radford, Ilya Chen, and Ilya Sutskever. Generating long sequences with sparse transformers. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[5] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[6] Shiyang Li et al. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. 2019. arXiv preprint.

[7] OpenAI. Gpt-2 model description. `https://huggingface.co/openai-community/gpt2`, 2023.

[8] Hugging Face. Parameter-efficient fine-tuning (peft) documentation. `https://huggingface.co/docs/peft/en/index`, 2023.

A Results

Method	Dev Accuracy (%)	Test Accuracy (%)
SST (Stanford Sentiment Treebank)		
Last-Linear	47.0	46.6
Full Model	50.8	54.8
CFIMDB (IMDB Sentiment Dataset)		
Last-Linear	87.8	—
Full Model	96.7	—

Table 2: Baseline accuracy for SST and CFIMDB.

B Spare Attention Results

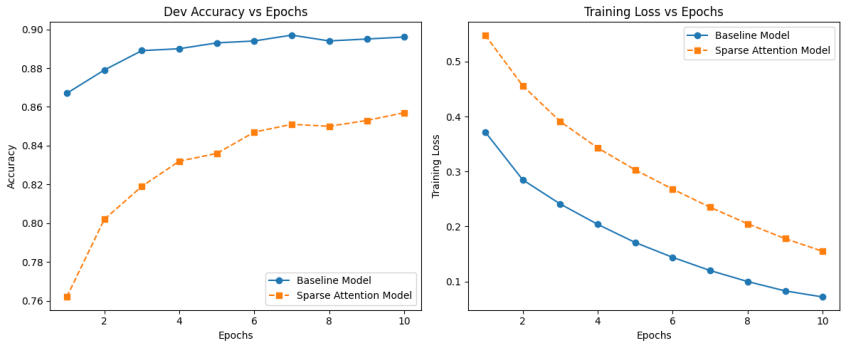
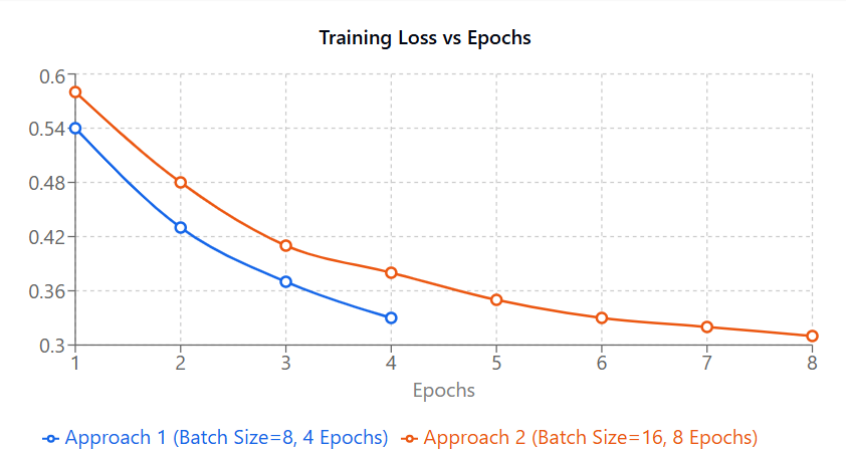


Figure 1: Training loss and accuracy over epochs for the baseline and sparse attention models on the paraphrase detection task

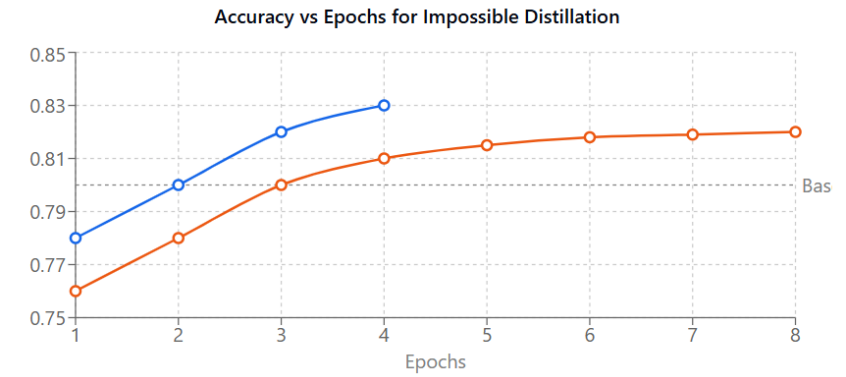
C Impossible Distillation Results

Method	Dev Accuracy (%)	Training Epochs
Quora Paraphrase Detection		
Baseline GPT-2	80.0	10
Approach 1	83.0	4
Approach 2	82.0	8

Table 3: Accuracy comparison for Impossible Distillation approaches on Quora dataset.



(a) Accuracy vs Epochs for Impossible Distillation



(b) Training Loss vs Epochs

Figure 2: Training metrics for Impossible Distillation approaches. Approach 1 achieves higher peak accuracy (0.83) after only 4 epochs, while Approach 2 extends to 8 epochs but reaches a slightly lower accuracy (0.82).

D Generative Pre-training Results

Configuration	CHRF Score (dev)
Self-Attention (Baseline)	40.517
LogSparse Attention	40.483
Self-Attention + Pre-Training	41.405
LogSparse Attention + Pre-Training	39.901

Table 4: CHFR Scores for Various Configurations

E Decoding Strategies Results

Decoding Strategy	Sonnet CHRF Score (dev)
Greedy	33.276
Top-k	41.095
Top-p	41.430
Top-p-k	40.684
Beam Search	26.778

Table 5: CHRF scores for different decoding strategies.

F Adaptation methods

Methods	Trainable Parameters	Avg GPU Memory Allocated
baseline	775,669,760 (100%)	12.78 GB
Freeze first half layers	118,064,640 (15.2209%)	4.77 GB
LoRA (r=4)	368,640 (0.0475%)	3.34 GB
LoRA (r=8)	737,280 (0.0950%)	3.35 GB
LoRA (r=16)	1,474,560 (0.1897%)	3.36 GB
LoRA (r=32)	2,949,120 (0.3788%)	7.55 GB

Table 6: Meta-data for different parameter efficient adaptation methods.

Methods	Final Training Loss	Final Perplexity	Sonnet CHRF Score (dev)
baseline	1.556	4.739	41.729
Freeze first half layers	3.533	34.215	40.434
LoRA (r=4)	4.260	70.834	40.083
LoRA (r=8)	4.250	70.072	39.675
LoRA (r=16)	4.239	69.347	40.516
LoRA (r=16) Epoch=40 (early stopped at 30)	3.948	51.822	38.484
LoRA (r=32)	4.253	70.310	40.546

Table 7: Performance for different parameter efficient adaptation methods.