Parameter Efficient BERT Fine-tuning

Stanford CS224N Default Project

Ang Li
Department of Computer Science
Stanford University
angli66@stanford.edu

Abstract

Recent years have witnessed the remarkable success of large language models (LLMs) in various natural language processing (NLP) tasks. As the size of LLMs increases, the resources required to train these models have skyrocketed, making it increasingly impractical to re-train LLMs for every sub-task. Consequently, fine-tuning pre-trained LLMs on downstream datasets in a cost-effective manner has become a critical area of research. Parameter-efficient fine-tuning (PEFT) methods aim to reduce the memory usage/running time required to finetune language models, while preserving performance. In this project, we explore the capabilities and limitations of two widely-used PEFT methods: Low-Rank Adaptation (LoRA) [1] and its recent variant, Weight-Decomposed Low-Rank Adaptation (DoRA) [2], on three downstream tasks including sentiment analysis, paraphrase detection and semantic textual similarity.

1 Key Information to include

- Mentor: Zhoujie Ding
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

The exponential growth and success of large language models (LLMs) in natural language processing (NLP) tasks have been nothing short of revolutionary. Models like BERT [3], GPT-3 [4], and their derivatives have set new benchmarks across a range of applications including machine translation, sentiment analysis, question answering, and more. The key to their success lies in their ability to capture intricate patterns and nuances in vast amounts of text data, leading to highly accurate and contextually aware outputs. However, this success comes at a substantial cost.

As the size and complexity of LLMs have increased, the computational resources required to train and fine-tune them also increased by a large margin. This includes significant memory consumption, prolonged training times, and the necessity for high-end hardware, which can be prohibitively expensive. Consequently, it has become increasingly impractical to re-train LLMs for every new sub-task, posing a major challenge for both researchers and practitioners.

Parameter-efficient fine-tuning (PEFT) methods have emerged as a promising solution to these issues. PEFT techniques aim to fine-tune only a subset of the model parameters or introduce additional trainable parameters, thereby reducing the computational burden while maintaining or enhancing model performance. This approach not only democratizes the fine-tuning process, making it accessible to a broader audience, but also facilitates rapid adaptation of pre-trained models to diverse applications without extensive retraining.

In this paper, we delve into two prominent PEFT methods: Low-Rank Adaptation (LoRA) [1] and its recent variant, Weight-Decomposed Low-Rank Adaptation (DoRA) [2]. LoRA reduces the number

of trainable parameters by decomposing the weight matrices into lower-dimensional forms, achieving parameter efficiency. While LoRA is lightweight and effective, it can fail to outperform finetuning the full model. To improve this, DoRA builds upon this concept by further decomposing the weight matrices to enhance model's performance while adding little runtime overhead.

We assess the effectiveness of these methods across three critical NLP tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Sentiment analysis involves determining the sentiment polarity of a given text, which has applications in areas like social media monitoring and customer feedback analysis. Paraphrase detection focuses on identifying whether two sentences convey the same meaning, which is essential for tasks such as plagiarism detection and question answering. Semantic textual similarity measures the degree of semantic equivalence between two texts, which is crucial for information retrieval and text summarization.

Current methods for fine-tuning LLMs, while effective, often fail to balance performance with efficiency. Traditional fine-tuning approaches require retraining large portions of the model, leading to high resource consumption. LoRA and DoRA present a viable alternative, promising substantial reductions in resource requirements without compromising on performance. Through our experiments, we aim to provide a comprehensive evaluation of LoRA and DoRA, highlighting their strengths and limitations.

3 Related Work

3.1 Transformer-Based Model

The Transformer model [5] revolutionized natural language processing (NLP) by leveraging self-attention mechanisms to process input sequences in parallel, rather than sequentially as in traditional recurrent neural networks. This architectural innovation enabled the modeling of long-range dependencies more effectively and efficiently, significantly improving performance on various NLP tasks.

Building upon the Transformer architecture, BERT (Bidirectional Encoder Representations from Transformers) [3] further advanced the field by introducing a bidirectional training approach. Unlike unidirectional models such as GPT [6], BERT processes text from both directions, capturing deeper and more nuanced contextual information. BERT's pre-training involves two objectives: Masked Language Modeling (MLM), where random words in a sentence are masked and the model learns to predict them, and Next Sentence Prediction (NSP), which helps the model understand the relationship between paired sentences. BERT's introduction marked a significant milestone in NLP, achieving state-of-the-art performance on a wide array of benchmarks at the time. Its success demonstrated the effectiveness of bidirectional pre-training in capturing complex language patterns and understanding context.

Following the success of BERT, several enhancements and variations have been proposed to further improve the performance and efficiency of Transformer-based models. RoBERTa [7] optimized BERT by removing the NSP objective, training on more data for longer periods, and using larger batch sizes, leading to improvements in several NLP tasks. SpanBERT [8] enhances BERT by focusing on predicting contiguous spans of text rather than individual tokens, which improves model's ability to capture and understand longer-term dependencies and relationships within the text.

Moreover, the GPT series [6, 9, 4] explored autoregressive training, where the model predicts the next word in a sequence, demonstrating impressive capabilities in generating coherent and contextually relevant text. GPT-3 [4], in particular, with 175 billion parameters, showcased the power of scaling up model size, achieving remarkable performance across diverse NLP tasks with few-shot or zero-shot learning.

These advancements underscore the rapid evolution of Transformer-based models and their significant impact on NLP. While BERT and its derivatives have set new performance standards, the increasing computational demands associated with training and fine-tuning such large models have prompted ongoing research into more efficient methodologies.

3.2 Parameter-Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) methods have been developed to address the significant computational and memory challenges associated with fine-tuning LLMs. Traditional fine-tuning involves updating a vast number of parameters, which can be resource-intensive and impractical for many applications. PEFT techniques aim to mitigate these challenges by reducing the number of trainable parameters, thereby decreasing the required computational resources and making the fine-tuning process more accessible and efficient.

One prominent PEFT method is Low-Rank Adaptation (LoRA) [1]. LoRA reduces the number of trainable parameters by decomposing the weight matrices of the model into lower-dimensional forms. Specifically, it introduces additional trainable matrices that are of lower rank, which are then combined with the original weight matrices during fine-tuning. This decomposition allows the model to retain most of its expressive power while significantly reducing the number of parameters that need to be updated. LoRA has demonstrated substantial efficiency gains, making it possible to fine-tune large models on resource-constrained hardware without a significant loss in performance.

Building on the principles of LoRA, Weight-Decomposed Low-Rank Adaptation (DoRA) [2] further enhances the fine-tuning process by improving the efficiency of parameter updates. DoRA focuses on decomposing the weight matrices to better handle the magnitude and direction of updates, which leads to more effective fine-tuning. This method allows for more precise updates, enhancing the overall performance of the model on downstream tasks without introducing additional computational costs.

PEFT methods, such as LoRA and DoRA, offer a compelling alternative to traditional fine-tuning approaches by balancing performance with efficiency. They enable the rapid adaptation of pre-trained models to specific tasks with significantly reduced resource requirements. As research in this area continues to evolve, further advancements and optimizations in PEFT methods are expected, paving the way for more sustainable and accessible fine-tuning of large language models.

4 Approach

4.1 Base Model

The base model we used for various fine-tuning experiments is the pre-trained Bidirectional Encoder Representations from Transformers (BERT) [3], a transformer-based model known for generating rich contextual word representations. Specifically, we implemented the minBERT model as specified in the default final project handout. MinBERT retains the core architecture of BERT but is optimized for the scope of our experiments. For a detailed description of its structure, refer to the project handout.

To effectively handle the three downstream tasks—sentiment analysis, paraphrase detection, and semantic textual similarity—concurrently, we attach three different linear layers after the [CLS] token of minBERT's output. These layers are tailored to the specific requirements of each task:

- **Sentiment Analysis**: The [CLS] token's representation is first passed through a dropout layer with a probability of 0.3 to prevent overfitting. Subsequently, it is fed into a linear layer with dimensions *HIDDEN SIZE* $\times$ *NUM CLASSES*, which outputs the final logits for sentiment classification.

- **Paraphrase Detection**: Both sentences in a pair are processed independently by minBERT to obtain their respective [CLS] token representations, denoted as [CLS1] and [CLS2]. These tokens are then concatenated and passed through a linear layer with dimensions 2 *HIDDEN SIZE* $\times 1$ to produce a single output logit, indicating whether the sentences are paraphrases.

- **Semantic Textual Similarity**: The pipeline for this task mirrors that of paraphrase detection. The [CLS] tokens from both sentences are concatenated and processed by a linear layer to produce a similarity score.

4.2 LoRA

Low-Rank Adaptation (LoRA) is a technique designed to reduce the memory usage and computational time required during fine-tuning while preserving model performance. The underlying hypothesis is that the changes needed to adapt a pre-trained model to a new task exhibit a low "intrinsic rank." For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA models the weight update $\Delta W \in \mathbb{R}^{d\times k}$ during fine-tuning as the product of two low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, where $r \ll \min(d, k)$. The final weight matrix is parameterized as $W_0 + BA$.

During fine-tuning, the original weight matrix W_0 is frozen, and only the low-rank matrices A and B, along with the final linear layers, are updated. This approach significantly reduces the number of parameters that need to be trained, thereby lowering the computational and memory requirements. The effectiveness of LoRA in various NLP tasks demonstrates its potential for efficient model adaptation without sacrificing accuracy.

As for implementation-level details, we replace linear layers that compute query, key and value in self-attention projections, dense linear layer after self-attention projections, and dense linear layers in intermediate and output projections in feed-forward network with LoRA layers of specific rank.

4.3 DoRA

While LoRA offers significant benefits in terms of efficiency, it may sometimes underperform compared to standard fine-tuning. To address this, Weight-Decomposed Low-Rank Adaptation (DoRA) proposes a refined approach that decomposes the magnitude and direction of weight updates to enhance fine-tuning performance without additional computational costs.

DoRA formulates the pre-trained weight matrix W in a manner that separates the magnitude and direction components. Specifically, the weight matrix W is expressed as:

$$W = m\frac{V}{\|V\|_c} = \|W\|_c\frac{V}{\|V\|_c}$$

where $m \in \mathbb{R}^{1\times k}$ is a learnable magnitude vector, $V \in \mathbb{R}^{d\times k}$ is a directional matrix, and $\|\cdot\|_c$ represents the vector-wise norm across each column. LoRA is applied to the directional component V.

To further enhance efficiency, the normalization factor $\frac{1}{\|V\|_c}$ is treated as a constant during back-propagation. This simplification reduces computational overhead while having minimal impact on performance. The fine-tuning process for DoRA mirrors that of LoRA, involving the freezing of the pre-trained weights and updating only the low-rank components and the final linear layers.

Similar to implementation details of LoRA, we replace the linear layers in self-attention module and bert layer with DoRA layers of specific rank.

5 Experiments

5.1 Data

For our experiments, we utilized three distinct datasets tailored to the specific requirements of each downstream task: sentiment analysis, paraphrase detection, and semantic textual similarity.

- **Sentiment Analysis**: For fine-tuning the sentiment analysis task, we used the Stanford Sentiment Treebank (SST) datasets. SST provides fine-grained sentiment labels for sentences, making it an ideal choice for training models to understand nuanced sentiments. The dataset consists of both binary and fine-grained labels, allowing us to evaluate the performance of our models on a comprehensive sentiment classification task.
- **Paraphrase Detection**: For the task of paraphrase detection, we employed the Quora Question Pairs dataset. This dataset contains pairs of questions from the Quora platform, labeled to indicate whether the questions are paraphrases of each other. The dataset is extensive and covers a wide range of topics, making it suitable for training models to identify semantic similarity and redundancy in natural language queries.

- **Semantic Textual Similarity**: To evaluate semantic textual similarity, we used the SemEval STS Benchmark Dataset. This dataset comprises sentence pairs annotated with similarity scores ranging from 0 to 5, where 0 indicates no semantic similarity and 5 indicates complete semantic equivalence. The STS Benchmark is a standard dataset for assessing the capability of models to capture semantic meaning and quantify the similarity between textual expressions.

Each dataset was pre-processed to ensure compatibility with the input requirements of the minBERT model. These datasets provided a diverse set of challenges for our fine-tuning experiments, allowing us to thoroughly evaluate the performance and efficiency of the LoRA and DoRA methods across different NLP tasks.

5.2 Evaluation method

For evaluating the performance of our models across the three downstream tasks, we employed different metrics tailored to the specific nature of each task:

- **Sentiment Analysis and Paraphrase Detection**: For these classification tasks, the primary evaluation metric is classification accuracy. Classification accuracy measures the proportion of correctly predicted instances among the total instances. It provides a straightforward and intuitive measure of performance, allowing us to gauge how well the model distinguishes between different classes. Higher accuracy indicates better model performance in correctly identifying sentiment polarity and detecting paraphrases.
- **Semantic Textual Similarity**: For this regression task, we used the Pearson correlation coefficient as the evaluation metric. The Pearson correlation coefficient measures the linear correlation between the ground truth similarity scores and the predicted similarity scores, ranging from -1 to 1. A coefficient of 1 indicates a perfect positive linear relationship, 0 indicates no linear relationship, and -1 indicates a perfect negative linear relationship. This metric is particularly suitable for semantic textual similarity tasks as it quantifies how well the predicted scores align with the actual semantic relationships between sentence pairs.

These evaluation metrics were chosen for their relevance and effectiveness in capturing the performance of models in their respective tasks. By using classification accuracy for sentiment analysis and paraphrase detection, and the Pearson correlation coefficient for semantic textual similarity, we ensure a comprehensive assessment of the model's capabilities across diverse NLP challenges.

Besides, we keep track of the training time and GPU memory usage of each run to compare runtime performance of the different methods.

5.3 Experimental Details

The training experiments were conducted on a single RTX 3060 GPU. Our experiments consisted of four parts: fine-tuning the last linear layers, fine-tuning the full model, fine-tuning with LoRA layers, and fine-tuning with DoRA layers. For all four experiments, we used a batch size of 16.

1. **Fine-Tuning Last Linear Layers**: In this setup, all layers except the last three linear layers for the three downstream tasks were frozen. We used a learning rate of 1×10^{-3} for this experiment.
2. **Fine-Tuning Full Model**: Here, gradients were allowed to backpropagate through all layers of the model. The learning rate was set to 1×10^{-5}.
3. **Fine-Tuning with LoRA Layers**: For LoRA, we chose a rank of 8 and a learning rate of 1×10^{-5}. All layers except the LoRA layers and the last three linear layers were frozen.
4. **Fine-Tuning with DoRA Layers**: Similarly, for DoRA, we used a rank of 8 and a learning rate of 1×10^{-5}. All layers except the DoRA layers and the last three linear layers were frozen.

To accommodate the three subtasks concurrently, for every gradient update, a batch was taken from each of the three datasets, and all batches were forward-passed through the model. Each batch size was 16. The length of an epoch was defined by the longest dataset among the three. The other

two shorter datasets were wrapped around to ensure they did not run out during an epoch. All four methods were trained for one epoch.

The decision to train for only one epoch was based on two observations:

- Post one epoch, the improvements were minimal, and there were signs of overfitting.
- The primary focus of this project was to study the performance-efficiency trade-off rather than solely enhancing performance. Therefore, the comparison of different methods under similar settings was emphasized.

5.4 Results

Table 1: Performance and Efficiency Metrics for Different Fine-Tuning Methods

Method	SA Acc	PD Acc	STS Corr	Memory Usage	Training Time
Last Linear	0.395	0.610	0.312	∼750 MB	∼30 min
Full Model	0.494	0.769	0.370	∼4100 MB	∼100 min
LoRA	0.307	0.632	0.239	∼800 MB	∼30 min
DoRA	0.320	0.632	0.235	∼950 MB	∼30 min

Table 1 contains the results of the experiment. The evaluation metrics are computed on the validation set. SA Acc refers to accuracy of sentiment analysis. PD Acc refers to accuracy of paraphrase detection. STS Corr refers to the correlation coefficient of semantic textual similarity. Memory usage and training time are recorded on a single RTX 3060 GPU.

6 Analysis

6.1 Fine-tuning Last Linear Layers vs. Fine-tuning Full Model

There is a significant overall improvement when transitioning from fine-tuning only the last linear layers to fine-tuning the full model. This result is intuitive: when only the last three linear layers are fine-tuned, the model is limited in its ability to extract and leverage common information across the three tasks. Each task is treated somewhat in isolation, preventing the model from effectively sharing learned representations.

In contrast, fine-tuning the full model allows gradients to propagate through all layers, enabling the model to transfer learned knowledge from one subtask to another. This holistic approach facilitates better integration and utilization of shared patterns and features, thereby enhancing overall performance across all tasks. The improvement observed underscores the importance of leveraging the entire model's capacity for fine-tuning, as it maximizes the benefits of transfer learning and shared representations.

However, as a trade-off, fine-tuning the full model requires significantly more GPU memory and training time. Even with our minBERT implementation, this approach incurs considerable overhead. In the context of large language models (LLMs), fine-tuning the full model might become infeasible due to these substantial resource demands.

6.2 PEFT

Comparing fine-tuning with LoRA to fine-tuning just the last linear layers or the full model, we observe that the performance of LoRA is slightly worse than fine-tuning the last linear layers only. However, it is important to note that this comparison may not be entirely fair. By replacing the linear layers in the self-attention module and BERT layers with LoRA layers, we lose the pre-trained information of the original layers. Consequently, these layers are reinitialized and trained from scratch, which can impact performance. Other potential reasons for the performance drop include the restricted representation power of the model due to the low-rank approximation of LoRA layers. Despite this, LoRA appears to be functioning correctly, as the loss continued to converge during training. Overall, it is reasonable to conclude that LoRA can still achieve relatively good performance, as evidenced by its superior paraphrase detection accuracy compared to fine-tuning only the last

linear layers. Additionally, adding LoRA to the model results in nearly identical runtime performance and GPU memory usage as fine-tuning the last linear layers, demonstrating LoRA's efficiency.

Switching from LoRA to DoRA provides a slight performance enhancement, with a 0.13 increase in sentiment analysis accuracy and a 0.04 decrease in semantic textual similarity correlation. The improvement is not very pronounced in our case, which may be due to the lack of pre-trained information and the fact that our fine-tuning datasets alone may not be sufficient to fully distinguish between the effectiveness of LoRA and DoRA. Nevertheless, the observed enhancement exists without any specific hyperparameter tuning compared to LoRA. It is noteworthy that changing from LoRA to DoRA incurs almost no runtime overhead and only a minimal increase in GPU memory usage. Thus, it is reasonable to conclude that replacing LoRA with DoRA can be beneficial, providing at least equivalent performance with minimal additional overhead.

6.3 Ablation Study on Rank

Table 2: Ablation Study on Rank of LoRA and DoRA

Method	**SA Acc**	**PD Acc**	**STS Corr**
LoRA (rank 8)	0.307	0.632	0.239
LoRA (rank 32)	0.311	0.632	0.237
DoRA (rank 8)	0.320	0.632	0.235
DoRA (rank 32)	0.322	0.632	0.230

Table 2 provides an ablation analysis on the rank of LoRA and DoRA. The results show that increasing the rank from 8 to 32 leads to minor changes in performance metrics for both LoRA and DoRA. For sentiment analysis, there is a slight improvement in accuracy with a higher rank for both methods. However, for paraphrase detection, the accuracy remains constant regardless of the rank, indicating that this task is less sensitive to changes in the rank of the low-rank matrices. For semantic textual similarity, a slight decrease in correlation is observed with a higher rank, suggesting a marginal trade-off in performance.

Overall, the analysis indicates that using a lower rank (such as 8) might be preferable for achieving better efficiency without substantial loss in performance, as the differences in performance metrics between ranks 8 and 32 are minimal.

7 Conclusion

In this paper, we explored the efficiency and performance trade-offs associated with various fine-tuning methods for large language models (LLMs), using minBERT as our base model. Our experiments compared the effectiveness of fine-tuning only the last linear layers, the full model, and incorporating Parameter-Efficient Fine-Tuning (PEFT) techniques such as LoRA and DoRA.

Fine-tuning the full model demonstrated significant performance improvements over fine-tuning just the last linear layers, highlighting the benefits of leveraging the entire model's capacity. However, this approach also incurred substantial increases in GPU memory usage and training time, which may render it infeasible for larger models in resource-constrained environments.

LoRA, designed to reduce memory and computational requirements by introducing low-rank adaptations, showed promising results in terms of efficiency. While its performance was slightly lower than that of fine-tuning the last linear layers, this can be attributed to the reinitialization of layers and the loss of pre-trained information. Despite these challenges, LoRA maintained efficient memory usage and training times comparable to fine-tuning the last linear layers, validating its potential as a resource-efficient fine-tuning method.

Transitioning from LoRA to DoRA provided a modest performance enhancement with minimal additional overhead. This indicates that DoRA can be a viable alternative to LoRA, offering slight improvements without significant increases in resource consumption.

In summary, our study underscores the importance of balancing performance and efficiency in fine-tuning LLMs. While full model fine-tuning delivers the best performance, PEFT methods like LoRA and DoRA present viable options for achieving competitive results with lower resource demands.

Future work could focus on optimizing these techniques further, exploring more sophisticated initialization strategies, and extending their application to other models and tasks.

8 Ethics Statement

In the rising era of AI, it is crucial to consider the ethical implications of new AI techniques. In the context of this project, which focuses on fine-tuning large language models (LLMs) at a lower cost, a potential risk is that it could enable individuals to imitate others' writing styles, raising ethical concerns. This issue is not unique to our project; it is a broader challenge associated with many LLMs. The ability to generate text that closely mimics human writing can be misused for purposes such as plagiarism, misinformation, and identity theft.

Additionally, the democratization of fine-tuning techniques might lead to the widespread creation and dissemination of harmful content, including biased, offensive, or misleading information. As researchers and practitioners, it is our responsibility to develop safeguards and guidelines to mitigate these risks. Studies on how to distinguish text written by humans from that generated by LLMs are essential and ongoing. Furthermore, transparency in the deployment of AI-generated content and adherence to ethical standards in AI research and application are paramount to ensure the responsible use of these powerful technologies.

References

[1] 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, 2021.

[2] 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. *arXiv preprint arXiv:2402.09353*, 2024.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[6] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[8] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans, 2020.

[9] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.