GPT2-TINYTINY: AN EXPLORATION AND IMPLEMENTATION OF GPT2 FROM SCRATCH

YUFA ZHOU [YUFAZHOU@SEAS], YIXIAO LING [LIING@SEAS], KEQI WU [KEQIWU@SEAS],

ABSTRACT. For this project we want to explore more deeply into the transformer architecture covered in lecture and homework, especially in GPT2 structure. We first implement the basic tiny version of GPT2 model from scratch, pretrain it on Wikitext dataset. Then we use some prompts to allow model to complete sentences for our evaluation. The results allowed us to observe the model's ability to capture intricate language patterns and nuances. In this project we have gained more hands-on experience of transformer models and deepened our understanding of attention mechanism and the pipeline of language problems.

1. INTRODUCTION AND BACKGROUND

This project is inspired by the introduction of the Transformer architecture in the paper "Attention is All You Need" by Vaswani et al. [12] in 2017. This architecture has become the foundation for many subsequent language models, such as BERT (Bidirectional Encoder Representations from Transformers) [1], GPT (Generative Pretrained Transformers) [9], and their iterations. In this project, we examined and focused mostly on GPT series models, which have revolutionized natural language processing (NLP) by enabling more effective and efficient handling of a wide array of tasks, such as question answering, text summarization, and without the need for task-specific architectural modifications. They have been pre-trained on vast amounts of text data and can be fine-tuned for specific NLP tasks, leading to state-of-the-art performance across various benchmarks.

BERT uses bidirectional context for word embeddings, meaning it learns information from both the left and right side of a token within a sentence. It's pre-trained on a large corpus and designed to understand the context of a word in search queries. However, GPT operates as an autoregressive model predicting subsequent words in a sentence, generating coherent and contextually relevant text from a given prompt; its unidirectional nature allows information utilization only from the past, not the future. GPT2 [10], an advancement over GPT1, is a more powerful language model with 1.5 billion to 175 billion parameters, enabling enhanced understanding and generation of human-like text. ChatGPT [7] is a variant of the GPT architecture, fine-tuned for conversational patterns using Reinforcement Learning from Human Feedback (RLHF). It's optimized to engage in dialogues and can provide responses that follow a conversation's flow, answer questions, simulate personalities, and even perform specific tasks like translation or calculation within the dialogue context.

Large Language Models (LLMs) is a rather heated topic nowadays and everyone use ChatGPT, GPT4, Claude etc. to help with their study and work. GPT based models have become an indispensable part of people's life in modern society. So we decide to further investigate and gain hands-on real experience with GPT technology to understand this game-changer instead of just behind the myth of this everyday life helper.

We first try to implement the basic structure of GPT2 with fewer parameter than the original paper thus called tinytiny version. We then use Wikitext-103 [5] dataset from Hugging face datasets library [4] to create the dataset that model use. And we pretrain the GPT2-tinytiny model with the dataset on our own GPU (GTX1660Ti) for 2 days to get the feeling of training a somewhat large model compared to the homework using not so powerful GPU and evaluate its results. We believe this can enhance our understanding of the difficulty of training a LLM and have bigger impression. Finally, we use some prompts to allow model to complete sentences and analyse the results.

Our code can be found on Github. [1]

1.1. Contributions.

- We preprocess Wikitext-103 dataset and implement tiny version of GPT2 from scratch.
- We pretrain the model on our own GPU to feel the difficulty of training LLM.
- We generate some samples from our pretrained model and analyse its results.

[1] https://github.com/MasterZhou1/gpt2-tinytiny

2. Approach

For this section we will talk about the details of how we implement the whole pipeline.

2.1. **Dataset:** For the dataset part, we mainly use dataset Wikitext-103 [5]. WikiText-103 is a comprehensive language dataset comprising over 100 million tokens extracted from verified "Good" and "Featured" articles on Wikipedia. Widely used in natural language processing tasks, it provides a diverse and extensive corpus for training and evaluating language models, particularly those based on transformer architectures like GPT.

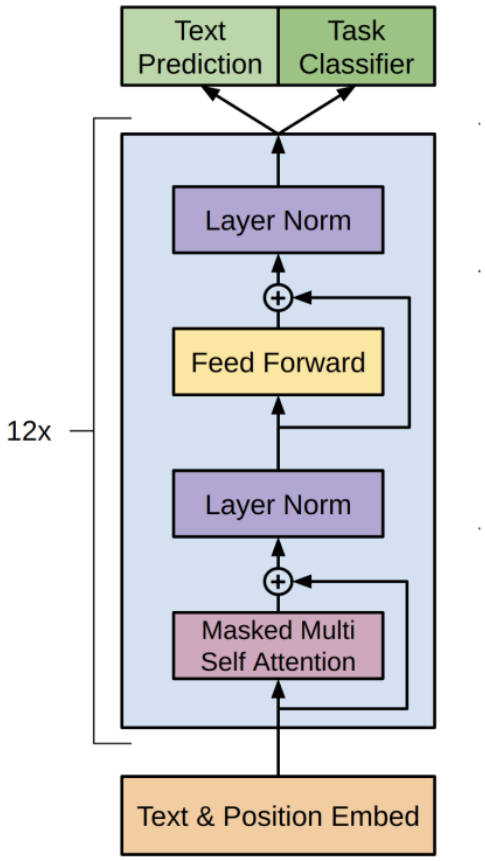

Figure 1. Transformer architecture used in GPT2

2.2. **Model:**

2.2.1. *Attention mechanism:* Attention mechanism can be thought as assigning different weights to different parts of the input data to compute the average weighted sum. This weighted sum (known as the context vector) is then used in subsequent layers of the neural network and then determine how much 'attention' is given to each input element. To see it in mathematical expression:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where

- Q is the Query matrix,
- K is the Key matrix,
- V is the Value matrix,

	n_layer	n_head	n_embd	block_size	bias	vocab_size	dropout	total parameters
Parameter:	4	4	384	256	False	50304	0.0	26.40m

TABLE 1. Hyperparameters for GPT2-tinytiny

- d_k is the Dimension of the keys (and queries), used for scaling.

In particular, GPT2 uses multiple 'heads' in its attention layers, allowing the model to simultaneously attend to information from different representation subspaces at different positions.
The mathematical equation is as follows:

$$\text{Multi-head Attention}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where W_i^Q, W_i^K, W_i^V, W^O are weight matrices for query, key, value, and output.

Additionally, there is a mask matrix that is a binary matrix applied during self-attention computation, where certain entries are masked to prevent attending to future tokens in a sequence, ensuring that each token can only attend to previous tokens and itself, preserving the autoregressive nature of the model.

2.2.2. *Embedding:* Embedding layer mapped each word to a unique vector into a high-dimensional space. Words with similar meanings are typically located close to each other in this space. GPT2 uses 2 kinds of embeddings, word embedding and positional embedding. For word embedding process, we just use GPT2 tonkenizer in huggingface to tokenize data and then use a nn.Embedding layer. Positional embeeding enable the model to understand the context and sequence of words in a text. In this implementation we use learned positional embedding (just nn.Embedding(block_size, n_embd) instead of fixed/absolute positional embedding in the transformer paper [12]. [2] Embedding process is fundamental for GPT's language understanding and generation capabilities, as it provides a nuanced representation of language data.

2.2.3. *GPT2:* GPT2 starts with embedding layers that transform input data to vectors that model can utilize. The central part is a repeated block of layers, stacked several times. Each block contains three components - Masked Multi-Self Attention, Layer norm, Feed Forward nerual network. Masked Multi-Self Attention component is responsible for the model's ability to focus on different parts of the input sequence when predicting each word. Layer norm helps stabilize the learning process by normalizing the output of the attention layers. The Feed Forward nerual network applies further transformations to the data before passing it to the next layer or block. At the top of the architecture, we use a simple linear layer as decoder that outputs logits for the next word in the sequence and it can also be adapted or replaced for specific tasks beyond text generation.

When implementing model, we use tying weight trick [8], which involves sharing the same set of parameters for both the input and output embeddings (word embedding layer and final linear decoder layer), allowing the model to efficiently learn relationships between words during training and reducing the overall number of parameters, which can enhance the model's generalization capabilities.

2.3. **Pretrain:** In the pretrain process, input dataset is a vast amount of raw unstructured data, containing a mix of high and low quality data. Using this dataset, GPT2 performs language modeling, predicting the next token in a sequence. This process teaches the model the probabilities of token sequences and enables it to understand and generate human-like text. After intensive computation, the result is a base model with a broad understanding of language patterns and nuances. This base model can then be further fine-tuned or deployed as is for various language tasks.

3. EXPERIMENTAL RESULTS

3.1. **Hyperparameters:** The parameter we use is described in table1. We select this small size of model to fit in the memory of our own GPU (GTX1660Ti 6GB memo). Parameters explanation:

- n_layer: number of transformer layers.
- n_head: number of heads in attention module.
- n_embd: number of embedding size in attention module.

[2]We believe that the descriptions of tips1 in huggingface GPT2 model docs https://huggingface.co/docs/transformers/model_doc/gpt2 is wrong. We check the original paper [10] and the source code in huggingface, and the implementation by them is indeed learned positional embedding not absolute embedding in the tips1.

- block_size: the context size of the model.
- bias: whether use bias in Linears and LayerNorms layer in transformer blocks.
- vocab_size: the vocabulary size of model.
- dropout: the dropout rate in transformer layers.

Some clarifications for hyperparameter: We choose n_layer, n_head, n_embd, block_size to be half size of GPT2-small in original paper. We didn't use bias in transformer blocks which is explained in the video [3] (the main code reference for this project) not much useful in practice. We choose a vocabulary size of 50304, which is the nearest multiple of 64 of 50257 (the GPT-2 tokenizer's vocabulary size), to enhance computational efficiency. Dropout for 0.0 in pretraining stage is a common practice. [4]

3.2. **Training:** During training we use random part of the dataset as batchs instead of training the dataset sequentially from the beginning. We randomly selects a batch of sequences from the specified split of the data, where each sequence has a length of block_size. Then we train it use pytorch's automatic mixed precision package to accelerate training. The training loss and validation can be seen in Fig4. We use AdamW optimizer and Cosine Annealing to change learning rate accordingly; otherwise model will stuck in the confusion zone taught in lecture. The final training loss is 3.648 and validation loss is 3.674. We have trained 3400 iterations with 1 iteration cost roughly 45s, so total time cost is approximately $45 * 3400/(24 * 60 * 60) = 1.77$ days.

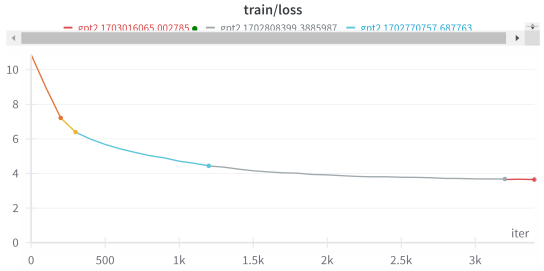

FIGURE 2. Training loss

FIGURE 3. Validation loss

FIGURE 4. Loss plot from wandb

3.3. **Results:** For results we test several prompts to test the performance of GPT2-tinytiny. We set temperature as 1.2 to add some randomness and only select parts of generated sentence to be shown as response.

Example 1:

- Prompt: Alice was friends with Bob. Alice went to visit her friend
- Response: Alice was friends with Bob. Alice went to visit her friend, David Jacobs. She later complained about Lee, as Lee sought employment rather than paying on such jobs. Betty received more scrutiny than doing so by a relative financial situation when Betty came up on a visit by the executives of Cameron, to get to the couple's final job before they were recording no other movies and agreed

From the first example we can see GPT2-tinytiny is capable of understand certain level of the context because it generates a proper human name after prompt, though the whole sentence is nonsense. Besides, the grammar of the sentence is also correct.

Example 2 (selected from the original text):

- Prompt: The Gambia women's national football team represents
- Response: The Gambia women's national football team represents the greatest college teams in the country. Since 1997, FIFA eligibility has adopted only the team's official name of current football because of their professional years. They are officially titled Maluasy Football Federation by president of the Portuguese East. Their players were named " Rhodes and Maguti " by the Portuguese as football and football

[3] https://www.youtube.com/watch?v=kCc8FmEb1nY&ab_channel=AndrejKarpathy

[4] We also try use 0.1 at the beginning but the loss is increasing. We thought this is reasonable because dropout may be counterproductive if the model is not overfitting. The effect of dropout will cause model hard to learn at the early stage of training.

- Original: The Gambia women 's national football team represents the Gambia in international football competition . The team , however , has not competed in a match recognised by FIFA , the sport 's international governing body , despite that organised women 's football has been played in the country since 1998 . The Gambia has two youth teams , an under @-@ 17 side that has competed in FIFA U @-@ 17 Women 's World Cup qualifiers , and an under @-@ 19 side that withdrew from regional qualifiers for an under @-@ 19 World Cup . The development of a national team faces challenges similar to those across Africa , although the national football association has four staff members focusing on women 's football .

The example 2 is selected from the original text. We can clearly see that the model is capable of memorize certain information in the dataset, though not all correct.

Example 3:

- Prompt: Philadelphia is in what U.S. state?
- Response: Philadelphia is in what U.S. state?, but because Coindroc served as " blackbird for the United States ", she believed afterwards Douglas declined to consider becoming a personal name. Regan accused Blunt from the local politician Jack Pattinson of Transvaore, because " although nothing's more like anything" is used, Smith and Lewis were

The example 3 is a question with question mark. The response is completely nonsense. From this output we can see that the model is not able to answer question yet. This is fair because we have not train the model to be specific at question answering problem and the Wikitext103 dataset we use only contain a small number of question mark. Thus the model perform badly on question ansering and needs to be further finetuned.

3.4. **Conclusion:** The results are pretty amazing for this tiny tiny size model to our opinion. We didn't expect our model to be able to generate perfect sentence because even ChatGPT can have many flaws. This size of model already have the ability of a low level understanding of context and generate grammar correct sentence. All in all, GPT is just a model that is trained to predict next token. No one knows the exact mechanism of how GPT understands the knowledge of dataset and generate desired response yet.

4. Discussion

Reflections:

- Training LLM is truly computational intensive. Before this project we haven't train a model that will literally cost days of training in GPU. Besides, the code complexity of GPT2 is a challenge and this experience really enhance our coding skill in pytorch.
- We know that the code is pretty simple and outcomes are not so good compared to the implementations made from big companies. But we learn a lot from this line-by-line code study and not just call some lib and run some good results. We learn something that can only be fully understood when you code it by your own hand. Examples are:
  - When implenmenting attention layer, why use nn.tril and softmax? Turns out that this is equivalent to the matrix multiplication form of average weighted summation using triangular matrix for masking. This form of implementation is a very clever way, enabling parallelism and more suitable for GPU computation.
  - Why are we using nn.multinomial to sample from the logits not using argmax like in homework? This is because if we use argmax model will be so deterministic and have many repetetitions when generating. If we use multinomial distribution sampling the model can have more creativity and we can also use a parameter called temperature to control the randomness here.
  - Why model generates new sequence based only on the logits of last time step? Won't it lose info of the former context? The model is trained on using all the previous context to predict the next token (attention mechanism), so the logits of last step token already contain the information of the previous context.
- Right now the model didn't generate good sentences that has knowledge. This might be caused by our small size of model and nature difficulty of generative model. And due to the sclaing law of LLM [3], increasing model size will significantly help model's performance but even more costly on time and computational resources.

Future works:

- If we have more time and computational resources, we want to explore more about the architectures and techniques used in LLMs such as BERT, LoRA [2], and Supervised Fine-Tuning (SFT) used in GPT2 (lora and SFT currently implemented but still have some flaws).

- We want to try bigger size of model, because the limited size of model used currently have a big gap from generating good sentences that make sense.
- We can use open-source pretrained LLMs like Bloom [6], LLaMA [11] and finetune those models to compare the results.

REFERENCES

[1] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[2] Edward J Hu et al. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021).

[3] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[4] Quentin Lhoest et al. “Datasets: A Community Library for Natural Language Processing”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 175–184. arXiv: 2109.02846 [cs.CL]. URL: https://aclanthology.org/2021.emnlp-demo.21.

[5] Stephen Merity et al. “Pointer Sentinel Mixture Models”. In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Byj72udxe.

[6] Niklas Muennighoff et al. “Crosslingual generalization through multitask finetuning”. In: *arXiv preprint arXiv:2211.01786* (2022).

[7] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[8] Ofir Press and Lior Wolf. “Using the output embedding to improve language models”. In: *arXiv preprint arXiv:1608.05859* (2016).

[9] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[10] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[11] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[12] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).