SuperAGI Assignment Report

Gaurav Kumar(2020CH10086)

CODING TASK 1

Brief Overview In this project, a small GPT-like model is implemented to perform text generation tasks on a limited dataset. The code demonstrates the construction and training of a transformer-based language model, starting with defining hyperparameters and preparing the dataset for training and validation. The model architecture includes **embedding layers, multi-head self-attention mechanisms, feed-forward layers, and layer normalization,** all crucial components of a transformer model. The training loop involves evaluating training and validation losses at intervals, visualized through plotted graphs. Finally, the model's text generation capability is showcased by generating a sequence of tokens based on a given context.

Model Architecture

Embeddings:
The code uses token embeddings, which turn each word into a vector of numbers. This makes it easier for the model to understand and process the text. I have implemented **Positional Embedding** part that helps the model know the position of each word in a sentence, which is important for understanding the order and context of words.

Transformer Layers:

Multi-Head Self-Attention Mechanism:
The model includes classes like head and **MultiHeadAttention**. These let the model focus on different parts of a sentence at the same time. It's like being able to read a sentence and pay attention to the beginning, middle, and end all at once, which helps understand the context and meaning better.

Point-Wise Feed-Forward Networks:
The FeedFoward class in the code is a part of the model that does extra processing on the data after the attention mechanism. It's like a second check to refine and better understand the information.

Loss Estimation Function:
There's a function called estimate_loss used to check how well the model is learning. It's like a mini-test for the model to see if it can predict the next word correctly. This helps in tuning the model and fixing errors during training.

I have implemented all this keep in mind avoiding use of in-built transformer libraries.

Sample Input

```
OpenAI is an artificial intelligence research laboratory and company that focuses on
developing and promoting artificial general intelligence (AGI). AGI refers to highly
autonomous systems that outperform humans at most economically valuable work.

OpenAI aims to ensure that AGI benefits all of humanity by adhering to a set of core
principles. These principles include broadly distributing the benefits of AGI,
prioritizing long-term safety, conducting research to make AGI safe, cooperating with
other institutions, and avoiding uses of AI that may harm humanity or concentrate
power.

The organization was founded in December 2015 as a non-profit, with the goal of
conducting research to advance AI capabilities and ensure its positive impact. In
2019, OpenAI transitioned to a for-profit company, stating that it would use any
influence it obtains over AGI deployment to ensure it is used for the benefit of all.

OpenAI has made significant contributions to the field of AI and natural language
processing. They have developed various language models, including the GPT (Generative
Pre-trained Transformer) series. GPT models are trained on massive amounts of data to
generate coherent and contextually relevant text. These models have been used in a
wide range of applications, including chatbots, content generation, language
translation, and more.

OpenAI also provides an API (Application Programming Interface) called OpenAI API,
which allows developers to access and utilize their language models. This enables
developers to integrate AI-powered capabilities into their applications and services.

OpenAI continues to engage in cutting-edge research, collaborating with the AI
community and working towards advancements in AGI technology. The organization's
mission revolves around ensuring the benefits of AGI are shared by all of humanity,
while addressing safety concerns and promoting responsible use of AI.
```

Sample Output

```
# generate from the model
context = torch.zeros((1, 1), dtype=torch.long, device=device)
print(decode(m.generate(context, max_new_tokens=1000)[0].tolist()))
```

```
OpenAI aims to ensure that AGI benefits all of humanity by adhering to
a set of core principles. These principles include broadly distributing
the benefits of AGI, prioritizing long-term safety, conducting research
to advance AI capabilities and enservices.

OpenAI continues to engage in cutting-edge research, collaborating with
the AGI ade sive imouncts and servi9 ofe, 19, therinclluations and more
2019, OpenAI transitioned to a for-profit company, stating that it woul
d use any influence it obtains over AGI deployment to ensure it is used
for the benefit of all.

OpenAI has made significant contributions to the field of AI and natura
l language processing. They have developed various language models, inc
luding the GPT (Generative Pre-trained Transformer) series. GPT models
are trained on massive amounnts of data to generate coherent and contex
tually relevant teOpenAI AIteneralled OpenAI API, which allows develope
rs to access and utilize their language models. This enables developers
to
```

Coding Task 2

Brief Overview The updated Python code represents an enhanced version of a GPT-2 style model, incorporating three key modifications: Rotary Positional Embedding, Group Query Attention, and Sliding Window Attention. These changes aim to experiment with and potentially improve the model's understanding of text context and relationships.

Model Architecture

Rotary Positional Embedding:

- Replaces the standard positional embeddings.
- Uses sine and cosine functions to encode the position of each word in a sequence, helping the model maintain the order of words.

Group Query Attention:

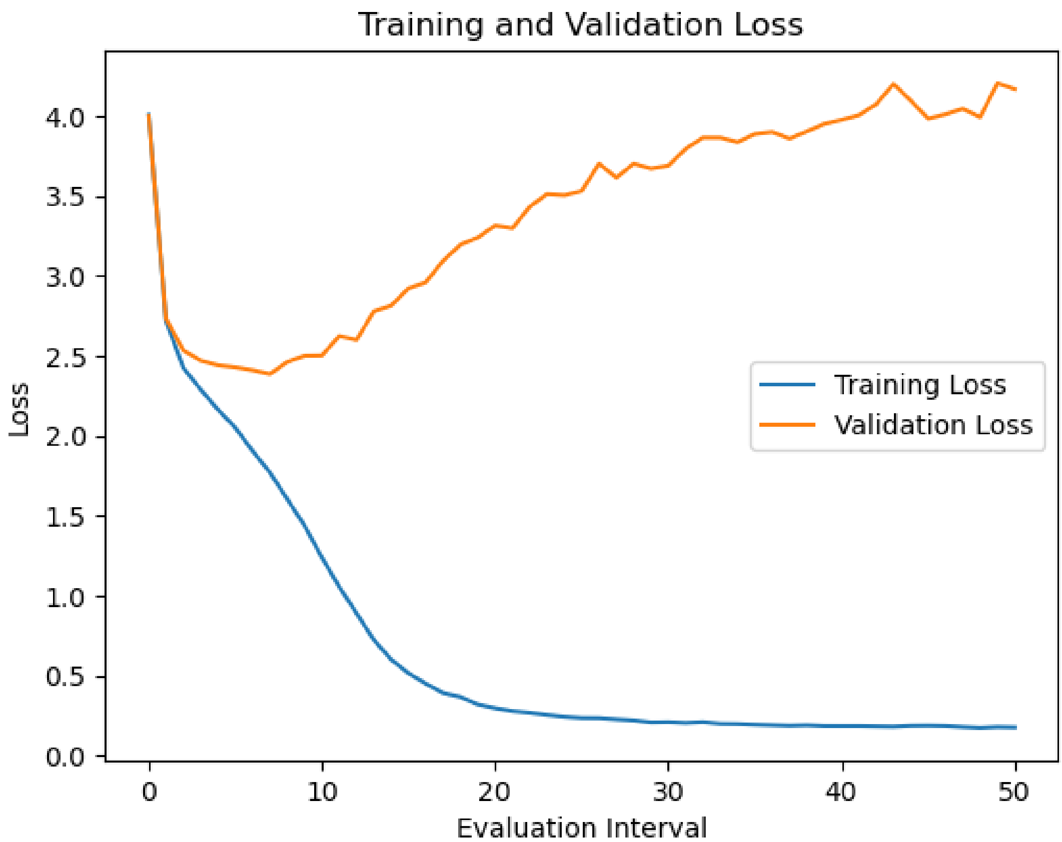
- Implemented in the **GroupQueryAttention class**.
- Divides the attention mechanism into groups, allowing the model to focus on different parts of the input separately. This can enhance the model's ability to capture nuanced relationships.

Sliding Window Attention:

- Added to control the scope of attention, limiting it to a fixed-size window around each word.
- This mechanism is expected to make the attention more focused and efficient, especially for longer sequences.

Model Size and Capabilities

- The model is relatively small, with a customizable number of layers and heads, allowing for adjustments in size and complexity.
- The embedding and attention mechanisms are scaled to the model's dimensions, maintaining a balance between performance and resource usage.



Challenges faced/Potential Pitfalls

Sample Output

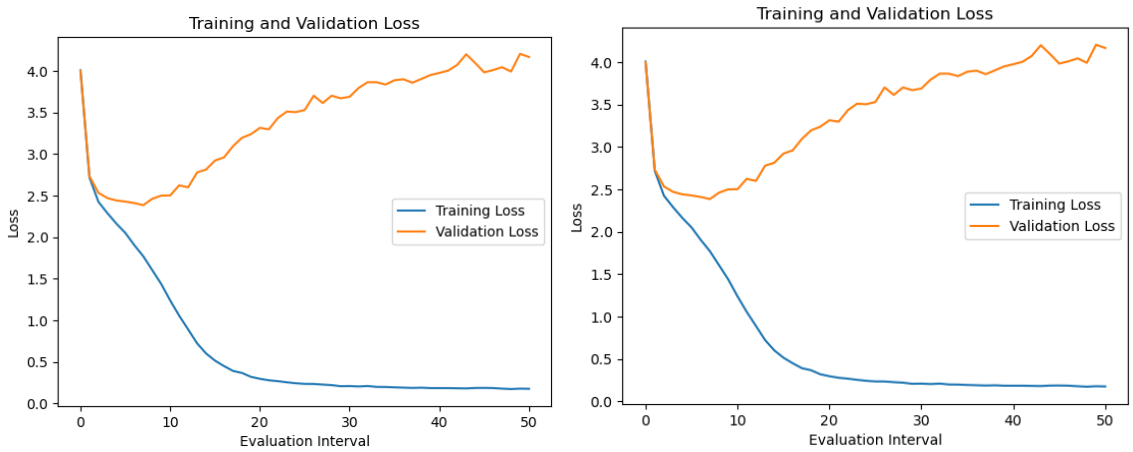
```
# generate from the model
context = torch.zeros((1, 1), dtype=torch.long, device=device)
print(decode(m.generate(context, max_new_tokens=1000)[0].tolist()))
```

```
Ope aimsoen, atiouwoutitutus thus, s, us, us, ingeding useling s, us, c
us, cedus, ce9, ccolucealucealuableses cedededelopopopopppoppppohvthop
h, copncompat, compand ampatimpramprampry mprmpmpmpmpmpmpmpmpmpmpmpmpmp
mpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmprmprmpmpmpmpmpmpmpmpmpmpmprmpmpmpmprm
prmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmprmprmpmpmprmprmprmpmpmprmpr
mprmpmprmpmprmpmpmpmpmpmpmpmpmpmpmpmpmpmprmpmpmpmpmpmpmpmpmprmpmpmpmpmp
mpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmprmp
mpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmprmprmprmpmpmpmpmpmpmpmpmpmp
mpmpmpmpmpmpmpmpmprmpmpmpmprmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmprmpmpmpmpmp
mpmpmpmpmpmpmpmpmprmprmpmpsmprmpmprmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmp
mpmpmpmpmpmpmpmprmprmpmpmpmprmpmpmpmpmpmpmpmpmpmpmpmpmprmpmprmpmpmpmprm
pmpmpmpmpmpmprmpmprmpmpmpmprmprmprmprmprmpmprmpmprmpmprmpmpmpmprmpmp
mpmpmpmpmpmpmpmpmpmpmpmpsmpmprmprmpmpmprmpmprmpmprmprmprmprmprmprmprmpr
mprmprmprmprmpmprmpmprmpmpmprmprmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpmpm
pm
```

- **Rotary Positional Embedding**: While it provides a more dynamic understanding of word positions, it may introduce complexity in learning patterns over very long sequences.
- **Group Query Attention**: Dividing attention into groups can improve focus but may risk losing some global context if not properly tuned.
- **Sliding Window Attention:** While it increases efficiency, there's a risk of missing long-range dependencies in the text.

Coding task 3

Single GPU training Loop for the model provide.



Part 2 and 3 (incomplete)
Given the hardware constraints of the M2 chip, it might be more practical to focus on single-device optimizations rather than trying to implement DDP or FSDP, which are more suited for multi-GPU environments.