Assignment 2 Report

I. Model architecture and parameters

The MiniGPT model is a simplified version of a Generative Pre-trained Transformer (GPT) model. It consists of the following main components:

1. **Token Embedding Table (token_embedding_table):** This layer converts input token IDs into dense vector representations (embeddings). The size of the embedding is D_MODEL.
2. **Position Embedding Table (position_embedding_table):** This layer provides positional information for the tokens in the sequence. It adds a learned embedding based on the token's position to the token embedding. The maximum sequence length is T.
3. **Transformer Blocks (blocks):** This is the core of the model, consisting of a sequence of N_LAYER TransformerBlocks. Each TransformerBlock contains:
   - **Multi-Head Attention (sa):** This layer allows the model to attend to different parts of the input sequence simultaneously. It uses N_HEAD parallel CausalSelfAttention heads. Each head has a size of D_MODEL // N_HEAD. A causal mask is applied to prevent attending to future tokens.
   - **Feed-Forward Network (ffwd):** A simple two-layer linear network with a ReLU activation.
   - **Layer Normalization (ln1, ln2):** Applied before the attention and feed-forward layers to help stabilize training.
4. **Final Layer Normalization (ln_f):** A layer normalization applied after the transformer blocks.
5. **Language Model Head (lm_head):** A linear layer that maps the output of the transformer blocks to the vocabulary size (VOCAB_SIZE) to produce logits for the next token prediction.

The key parameters defining the model's size and training configuration are:

- **B (Batch Size):** 16 - The number of sequences processed in parallel during training.
- **T (Sequence Length):** 64 - The maximum length of input sequences the model can handle.
- **D_MODEL (Embedding Dimension):** 128 - The size of the token and positional embeddings, and the hidden size throughout the transformer blocks.
- **N_HEAD (Number of Attention Heads):** 4 - The number of parallel attention heads in the MultiHeadAttention layer.
- **N_LAYER (Number of Transformer Layers):** 2 - The number of TransformerBlocks in the model.
- **LEARNING_RATE:** 5e-4 - The learning rate for the AdamW optimizer.
- **NUM_EPOCHS:** 5 - The number of times the training loop will iterate over the entire dataset.
- **VOCAB_SIZE:** 30522 - The size of the vocabulary, determined by the tokenizer (bert-base-uncased).

In essence, the MiniGPT model processes input sequences by embedding tokens and their positions, then applying multiple layers of self-attention and feed-forward networks to learn relationships between tokens and predict the next token in the sequence.

II. Dataset details

For dataset, we use the dataset collected and processed from Assignment 1, Wikipedia English 20231101.en, the first 150k rows, about 1.14G data.

III. Training setup and hyperparameter experiments

Training Setup:

1. **Device:** The code attempts to use a GPU (cuda) if available, otherwise it falls back to the CPU (cpu).
2. **Dataset:** A custom PyTorch Dataset called NextTokenPredictionDataset is used to load data from a processed dataset stored locally.
3. **DataLoader:** A DataLoader is used to iterate over the dataset in batches. A custom_collate_fn is used to handle padding of sequences to a maximum length of T + 1. The input X is the sequence up to the second to last token, and the target Y is the sequence shifted by one token to the right.
4. **Optimizer:** The AdamW optimizer is used for training.
5. **Loss Function:** Cross-entropy loss is used for the next token prediction task.
6. **Metrics:** The code tracks and reports the average training loss and perplexity at specified intervals. Perplexity is calculated as the exponentiation of the loss.
7. **Checkpointing:** The model's state dictionary is saved to a file (mini_gpt_checkpoint.pt) after each epoch.
8. **Metric Saving:** Training losses and perplexities are saved to a .npz file (training_metrics.npz) for later visualization.

Hyperparameters:

The following hyperparameters are defined and used in the training process:

- **B (Batch Size):** 16
- **T (Sequence Length):** 64
- **D_MODEL (Embedding Dimension):** 128
- **N_HEAD (Number of Attention Heads):** 4
- **N_LAYER (Number of Transformer Layers):** 2
- **LEARNING_RATE:** 5e-4
- **NUM_EPOCHS:** 5
- **VOCAB_SIZE:** 30522 (derived from the tokenizer)
- **EVAL_INTERVAL:** 100 (Although defined, evaluation is not explicitly implemented in the provided training loop; this variable seems to be a placeholder or for future use).
- **LOG_INTERVAL:** 10 - The frequency (in steps) at which training loss and perplexity are logged.
- **CHECKPOINT_PATH:** "mini_gpt_checkpoint.pt" - The file path for saving model checkpoints.

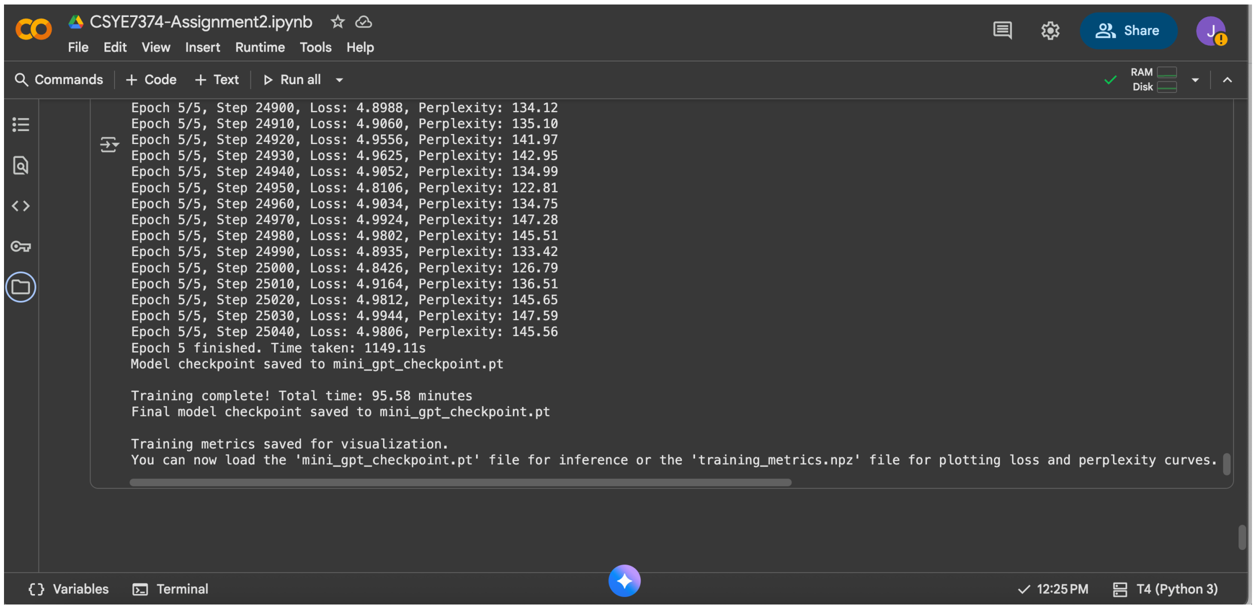
The training loop iterates for a fixed number of epochs, processing data in batches, calculating the loss, performing backpropagation, and updating the model's weights using the optimizer. Training progress is logged periodically, and checkpoints are saved after each epoch.

IV. Observations and challenges

I deliberately used both GPU on Colab and CPU on my own MacBook to train. The difference is striking. Despite the powerful M1Max chip, it still took 800 minutes to train locally, while it took only less than two hours to train on GPU, thanks to its significant parallelism.

Also, since the vectors are initiated randomly each time, the training outcome is different for each session. Notice that in the following images, the loss and perplexity are lower on Colab compared to local computer.

On Colab

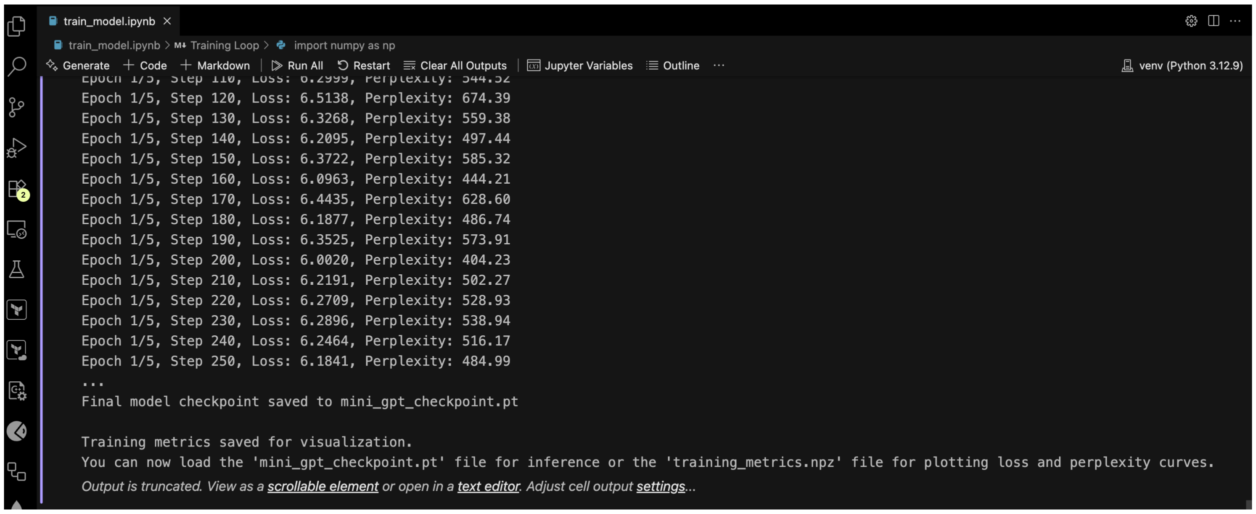

CSYE7374-Assignment2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Variables Terminal
12:25 PM T4 (Python 3)

```
Epoch 5/5, Step 24900, Loss: 4.8988, Perplexity: 134.12
Epoch 5/5, Step 24910, Loss: 4.9060, Perplexity: 135.10
Epoch 5/5, Step 24920, Loss: 4.9556, Perplexity: 141.97
Epoch 5/5, Step 24930, Loss: 4.9625, Perplexity: 142.95
Epoch 5/5, Step 24940, Loss: 4.9052, Perplexity: 134.99
Epoch 5/5, Step 24950, Loss: 4.8106, Perplexity: 122.81
Epoch 5/5, Step 24960, Loss: 4.9034, Perplexity: 134.75
Epoch 5/5, Step 24970, Loss: 4.9924, Perplexity: 147.28
Epoch 5/5, Step 24980, Loss: 4.9802, Perplexity: 145.51
Epoch 5/5, Step 24990, Loss: 4.8935, Perplexity: 133.42
Epoch 5/5, Step 25000, Loss: 4.8426, Perplexity: 126.79
Epoch 5/5, Step 25010, Loss: 4.9164, Perplexity: 136.51
Epoch 5/5, Step 25020, Loss: 4.9812, Perplexity: 145.65
Epoch 5/5, Step 25030, Loss: 4.9944, Perplexity: 147.59
Epoch 5/5, Step 25040, Loss: 4.9806, Perplexity: 145.56
Epoch 5 finished. Time taken: 1149.11s
Model checkpoint saved to mini_gpt_checkpoint.pt

Training complete! Total time: 95.58 minutes
Final model checkpoint saved to mini_gpt_checkpoint.pt

Training metrics saved for visualization.
You can now load the 'mini_gpt_checkpoint.pt' file for inference or the 'training_metrics.npz' file for plotting loss and perplexity curves.
```

On vscode


train_model.ipynb
train_model.ipynb > Training Loop > import numpy as np
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
venv (Python 3.12.9)

```
Epoch 1/5, Step 110, Loss: 6.2999, Perplexity: 544.52
Epoch 1/5, Step 120, Loss: 6.5138, Perplexity: 674.39
Epoch 1/5, Step 130, Loss: 6.3268, Perplexity: 559.38
Epoch 1/5, Step 140, Loss: 6.2095, Perplexity: 497.44
Epoch 1/5, Step 150, Loss: 6.3722, Perplexity: 585.32
Epoch 1/5, Step 160, Loss: 6.0963, Perplexity: 444.21
Epoch 1/5, Step 170, Loss: 6.4435, Perplexity: 628.60
Epoch 1/5, Step 180, Loss: 6.1877, Perplexity: 486.74
Epoch 1/5, Step 190, Loss: 6.3525, Perplexity: 573.91
Epoch 1/5, Step 200, Loss: 6.0020, Perplexity: 404.23
Epoch 1/5, Step 210, Loss: 6.2191, Perplexity: 502.27
Epoch 1/5, Step 220, Loss: 6.2709, Perplexity: 528.93
Epoch 1/5, Step 230, Loss: 6.2896, Perplexity: 538.94
Epoch 1/5, Step 240, Loss: 6.2464, Perplexity: 516.17
Epoch 1/5, Step 250, Loss: 6.1841, Perplexity: 484.99
...
Final model checkpoint saved to mini_gpt_checkpoint.pt

Training metrics saved for visualization.
You can now load the 'mini_gpt_checkpoint.pt' file for inference or the 'training_metrics.npz' file for plotting loss and perplexity curves.
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...