Assignment 2 Report — Mini GPT Training

1. Objective

The goal of this assignment is to implement and train a mini GPT-style Transformer language model for next-token prediction. The project includes tokenization, dataset preparation, model implementation, training, evaluation using loss and perplexity, and visualization of learning progress.

2. Dataset and Preprocessing

I used the Wikipedia text dataset loaded via the HuggingFace datasets library, consistent with the dataset used in Assignment 1.
To ensure data quality, each text sample was cleaned by removing empty, invalid, or low-content entries.
After cleaning, the text was tokenized using the GPT-2 tokenizer. The tokenized sequences were segmented into fixed-length blocks to support next-token prediction training. This preprocessing step allows the model to learn language patterns in a structured and memory-efficient manner.
Tokenization Process:

- Tokenizer: GPT-2 (AutoTokenizer)
- Padding token set to EOS token
- Each text sample was converted into token ID sequences

Block Construction”

- block_size = 128
- Each block forms a training sample
- Input = first 127 tokens
- Target = next 127 tokens (next-token prediction)

3. Model Architecture

We implemented a MiniGPT Transformer model, consisting of:

- Token embedding layer
- Positional embedding layer
- Multi-head self-attention
- Feedforward layers
- Residual connections and layer normalization
- Final linear projection to vocabulary space

Hyperparameter:
Vocabulary size: GPT-2 vocab
Embedding dimension: 128
Transformer layers: 2
Attention heads: 4
Sequence length: 127
Optimizer: AdamW
Learning rate: 1e-3
Epochs: 2

4. Training Procedure

Training was performed using next-token prediction:

- Loss function: Cross-Entropy Loss
- Back propagation: enabled
- Optimizer: AdamW
- Device: Apple MPS GPU (Metal backend)

During training, both loss and perplexity were tracked per epoch.

5. Evaluation Metrics

Loss:
Loss measures how well the predicted token probabilities match ground truth.
Perplexity (PPL):
Perplexity is computed as:

$$PPL = e^{Loss}$$

It represents how uncertain the model is when predicting the next token.
Lower perplexity indicates better language modeling performance.

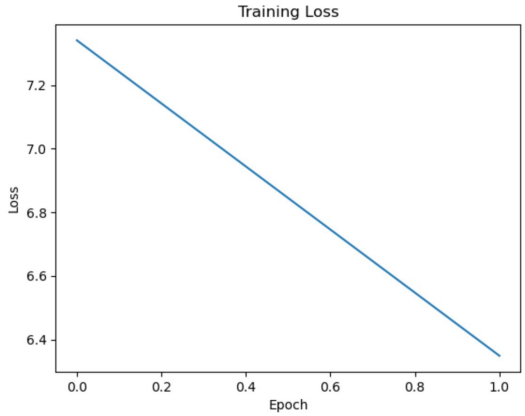
6. Results

Training Metrics:

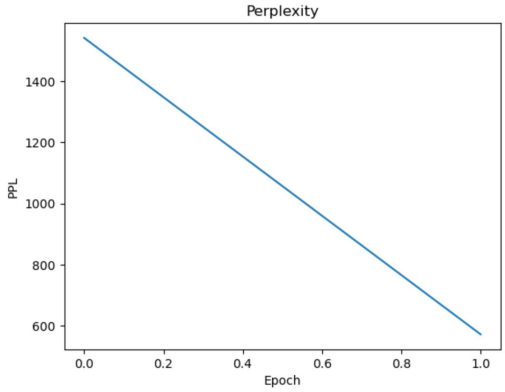
Epoch	Loss	Perplexity
1	7.34	1541.69
2	6.35	571.70

7. Visualizations

Training Loss Curve



Perplexity Curve



Both curves show a clear downward trend, demonstrating stable learning behavior.

8. Analysis and Discussion

The decreasing perplexity confirms that the Transformer model learned contextual dependencies.
Training for more epochs or increasing model depth could further improve performance
Larger datasets or longer sequence lengths may enhance language fluency.