Building a Small-Scale Foundation Model (Mini-GPT) from Scratch

(Submitted by: Ahsan Zafar H. Syed, NUID: 002801441)

1. Model Architecture and Parameters

Overview

- Embedding dimension: 128
- Transformer layers: 2
- Multi-head attention heads: 4
- Feed-Forward hidden size: 512
- Dropout: 0.1
- Maximum sequence length: 64
- Vocabulary size: ~50k

Architectural Components

Token Embedding Layer
Maps token IDs to dense vectors of size 128.

Positional Embedding
A learnable positional embedding is added to token embeddings to encode word order.

Self-Attention Blocks
Each block includes:

- LayerNorm
- Multi-Head Attention (causal masked)
- Feed-Forward MLP
- Residual connections

Causal Mask
Ensures the model predicts only from past tokens.

Output Projection
A linear layer maps hidden states to logits over the vocabulary for next-token prediction.

Parameter Count

Approximately 1–2 million parameters, depending on vocabulary and embedding size. Suitable for CPU/GPU training in a classroom environment.

2. Dataset Details

Source of Dataset

The dataset used here is exactly the preprocessed output from Assignment 1. It consists of:

- Cleaned raw text (~1.5GB)
- Tokenized and chunked sequences using the GPT-2 Byte Pair Encoding tokenizer
- Token blocks saved as token_blocks.pt
- Vocabulary size: ~50k BPE tokens

Sequence Construction for Training

To train a next-token prediction objective, the flattened token stream was converted into overlapping sequences:

- Input (x): tokens [i : i+SEQ_LEN]
- Target (y): tokens [i+1 : i+1+SEQ_LEN]
- Sequence length used: 64 tokens (valid range: 32–128 per requirements)

Train/Validation Split

- 90% for training
- 10% for validation
- Both splits shuffled and batched efficiently through PyTorch DataLoader

3. Training Setup and Hyperparameter Experiments

Task

The model was trained on a next-token prediction objective:

$$\text{" loss} = -t \sum \log p(x_{t+1} \mid x_{\le t}) \text{ "}$$

Loss & Optimizer

- Loss function: CrossEntropyLoss
- Optimizer: AdamW
- Learning rate: 3e-4
- Batch size: 32
- Epochs: 5

Perplexity Metric

Perplexity was computed as: “ $PPL = e^{loss}$ “

This indicates how confidently the model assigns probability mass to the next token.

Hyperparameter Experiments

Embedding Dimension:

Embedding Dimension	Effect
64	Faster, lower capacity
128	Recommended baseline
256	More expressive but slower

Number of Layers:

Layers	Effect
1	Learns basic structure; limited context modeling
2	Better perplexity; best efficiency/ performance balance

Learning Rate:

- 1e-3 → unstable spikes
- 5e-4 → better
- 3e-4 → smoothest convergence (used final)

Batch Size:

- 16 → noisy gradients
- 32 → stable
- 64 → memory constraints on some GPUs

4. Observations and Challenges

Dataset Size

Flattening a ≈1GB token stream required careful memory management.

Sequence Packing

Choosing SEQ_LEN=64 balanced:

- Information content
- GPU/CPU memory
- Training speed

Causal Masking

Attention mask errors were the most common debugging issue.

Overfitting vs Underfitting

With only 1–2 layers, underfitting is common, but this is expected and acceptable.

Training Compute

Training on CPU is slow; GPU strongly recommended.