Assignment 2: Building a Small-Scale Foundation Model from Scratch

Student Name: Chen Ting Kuo **Course:** CSYE7374 **Date:** February 12, 2026

---

1. Model Architecture and Parameters

We implemented a decoder-only transformer language model (Mini-GPT) in PyTorch, following the GPT-2 architectural design. The model is composed of the following components:

Token and Positional Embeddings Each input token ID is mapped to a 256-dimensional vector via a learned token embedding table. A separate learned positional embedding of the same dimension is added to encode the position of each token within the sequence (positions 0–127). This allows the model to distinguish tokens based on both their identity and their location in the sequence.

Transformer Blocks (× 2) The model stacks 2 identical transformer blocks. Each block follows a Pre-LayerNorm design:

- Layer Normalization → Causal Self-Attention → Residual Connection
- Layer Normalization → MLP (Feed-Forward Network) → Residual Connection

Causal Self-Attention Multi-head attention with 4 heads, each operating on 64 dimensions (256 / 4 = 64). A lower-triangular causal mask ensures each token can only attend to itself and preceding tokens, enforcing the autoregressive property required for next-token prediction.

MLP A two-layer feed-forward network that expands the hidden dimension by 4× (256 → 1024) with GELU activation, then projects back to 256. This allows the model to learn complex non-linear transformations per token.

Output Head A linear layer projects from 256 dimensions to 50,257 (vocabulary size), producing logit scores for next-token prediction. The output head shares weights with the token embedding table (weight tying), which reduces parameter count and improves generalization.

Model Configuration Summary

Hyperparameter	Value
Vocabulary Size	50,257
Embedding Dimension (`n_embd`)	256
Attention Heads (`n_head`)	4
Transformer Layers (`n_layer`)	2
Sequence Length (`block_size`)	128
Dropout	0.1
Total Parameters	~30M

2. Dataset Details

The training data was carried over directly from Assignment 1, which collected and preprocessed over 1 GB of multi-domain English text across three domains.

Data Sources

Domain	Dataset	Documents
News	`cc_news` (Jan 2017 – Dec 2019)	100,000
Encyclopedic	`wikimedia/wikipedia` (20231101.en)	100,000
General Web	`Skylion007/openwebtext`	100,000

Preprocessing Pipeline (Assignment 1) Raw text was cleaned via MD5-based exact deduplication (16,358 duplicates removed), lowercasing, whitespace normalization, and removal of documents shorter than 50 words (9,921 removed). The final cleaned dataset contained 273,721 documents (1.10 GB).

Text was tokenized using the GPT-2 Byte-Level BPE tokenizer (vocabulary size: 50,257) and stored as sharded PyTorch tensors (`shard_0.pt` – `shard_5.pt`), each containing approximately 50,000 blocks of 1,024 tokens, totaling approximately 266 million tokens.

Data Loading for Assignment 2 Since Assignment 2 requires a sequence length of 32–128 tokens, each 1,024-token shard row was flattened and re-chunked into blocks of 129 tokens (128 input + 1 target offset). A custom `GPTDataset` (PyTorch `IterableDataset`) loads one shard at a time to avoid RAM overflow and applies two-level shuffling: shard-order shuffling at the start of each epoch and row-level shuffling within each shard.

---

3. Training Setup and Hyperparameter Configuration

Training Loop The model was trained using next-token prediction with cross-entropy loss. At each step, the model receives an input sequence `x` of 128 tokens and predicts the next token at every position simultaneously, compared against target sequence `y` (x shifted right by one position).

Optimizer and Scheduler We used AdamW (Adam with decoupled weight decay) as the optimizer, with a cosine annealing learning rate scheduler that gradually reduces the learning rate from the initial value toward zero over the course of training. Gradient clipping (max norm = 1.0) was applied at every step to prevent gradient explosion.

Training Configuration

Setting	Value
Optimizer	AdamW
Learning Rate	3e-4
Weight Decay	0.01
Batch Size	16
Epochs	15
Steps per Epoch	2,000
Gradient Clip	1.0
LR Scheduler	Cosine Annealing
Checkpoint	`checkpoint_02.pt`

Perplexity Perplexity was computed at the end of each epoch as `exp(average_loss)`. It represents how many tokens the model is effectively choosing between at each prediction step — lower is better. A random model over 50,257 tokens would have a perplexity of 50,257; a well-trained small model typically reaches the hundreds to low thousands range.

Experiment Tracking All training metrics (per-step loss, per-step learning rate, per-epoch average loss, and perplexity) were logged to Weights & Biases (wandb) under the project `mini-gpt-csye7374`, enabling real-time monitoring of training dynamics.

---

4. Observations and Challenges

Observations Training loss decreased steadily across epochs, confirming that the model was learning to

model token distributions from the dataset. The cosine learning rate schedule helped stabilize convergence in later epochs, preventing oscillation around the minimum. The multi-domain nature of the dataset (news, encyclopedic, web text) provided diverse training signal, which is important for preventing the model from overfitting to a single writing style.

Weight tying between the token embedding and output head proved effective — it reduces the total parameter count while enforcing consistency between how the model represents input tokens and how it scores output tokens.

Challenges

Memory Management: Loading all shard files simultaneously would exceed available RAM. This was resolved by the streaming `IterableDataset` design, which keeps only one shard (~50–100 MB) in memory at any time.

Sequence Length Mismatch: Assignment 1 tokenized data into 1,024-token blocks, while Assignment 2 requires 32–128 tokens. Re-chunking was necessary and handled inside the data loader without re-running the full tokenization pipeline.

Lowercasing Trade-off: The aggressive lowercasing applied during Assignment 1 preprocessing reduces vocabulary complexity but loses semantic distinctions such as proper nouns (e.g., "Apple" the company vs. "apple" the fruit). For a production-scale model, a cased tokenizer would be preferable.

Small Model Capacity: With only 2 transformer layers and a 256-dimensional embedding, the model has limited capacity to capture long-range dependencies. Perplexity remains relatively high compared to large-scale models, which is expected given the architectural constraints of this assignment.
