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

Mini-GPT Language Model Implementation and Hyperparameter Analysis

Student Name: [Your Name]

1. Model Architecture and Parameters

This project implements a Mini-GPT transformer-based language model for next-token prediction. The model follows the standard GPT architecture with multi-head self-attention and feedforward layers.

1.1 Architecture Components

- **Token Embedding Layer:** Converts input tokens to dense vector representations
- **Positional Encoding:** Learned positional embeddings to capture sequence order
- **Transformer Blocks:** 2 layers (baseline), each containing multi-head self-attention (4 heads), layer normalization, feedforward network with GELU activation, and residual connections
- **Output Layer:** Linear projection to vocabulary size (50,257 tokens)

1.2 Model Configurations

Parameter	Baseline (Exp 1)	Best Model (Exp 3)
Vocabulary Size	50,257	50,257
Embedding Dimension	128	256
Attention Heads	4	4
Transformer Layers	2	2
Total Parameters	13.3M	27.3M

The best model uploaded to: https://huggingface.co/Avvvvva/mini-gpt-foundation-model

2. Dataset Details

The model was trained on preprocessed text data from Assignment 1, consisting of tokenized blocks from multiple sources.

2.1 Data Sources

- **WikiText:** 189 blocks
- **OpenWebText:** 312,366 blocks
- **AG News:** 0 blocks

2.2 Data Processing

- **Total Samples:** 312,555 tokenized blocks
- **Sequence Length:** 128 tokens per input block
- **Train/Val Split:** 90% training (281,299 samples) / 10% validation (31,256 samples)

- **Tokenizer:** GPT-2 BPE tokenizer
- **Batch Processing:** Dynamic batching with shuffling for efficient training

3. Training Setup and Hyperparameter Experiments

Five experiments were conducted to analyze the impact of different hyperparameters on model performance. All models were trained for 2 epochs with proper train/validation split.

3.1 Training Configuration

- **Loss Function:** Cross-entropy loss for next-token prediction
- **Optimizer:** AdamW with gradient clipping (max_norm=1.0)
- **Evaluation Metric:** Perplexity = exp(cross-entropy loss)
- **Training Device:** CUDA GPU (T4)
- **Validation:** Full validation set evaluation after each epoch, with step-level sampling every 500 steps
- **Checkpointing:** Model saved after each epoch with complete training history to Google Drive

3.2 Hyperparameter Experiments

Experiment	Configuration	Val Perplexity	Rank
Exp 1: Baseline	embed=128, layers=2, lr=5e-4, bs=32	160.61	3rd
Exp 2: Higher LR	embed=128, layers=2, lr=1e-3, bs=32	157.59	2nd
Exp 3: Larger Embed	embed=256, layers=2, lr=5e-4, bs=32	122.75	**1st (Best)**
Exp 4: Fewer Layers	embed=128, layers=1, lr=5e-4, bs=32	175.23	5th
Exp 5: Larger Batch	embed=128, layers=2, lr=5e-4, bs=64	166.28	4th

4. Results and Analysis

4.1 Individual Experiment Results

Each experiment was trained with step-level monitoring, logging metrics every 500 steps (250 steps for Experiment 5). The following figures show the training and validation loss/perplexity curves for each experiment.

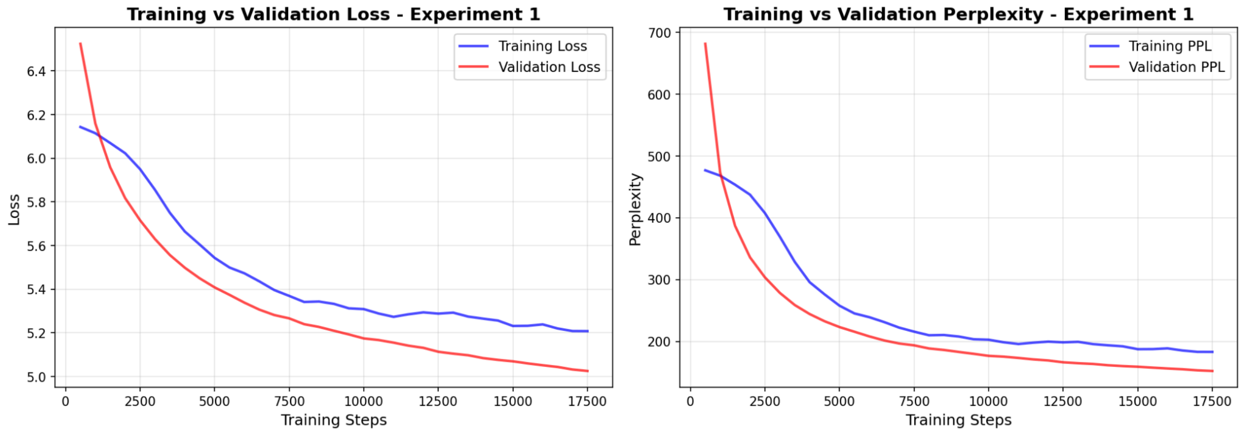

Figure 1: Training and validation loss/perplexity curves for Experiment 1

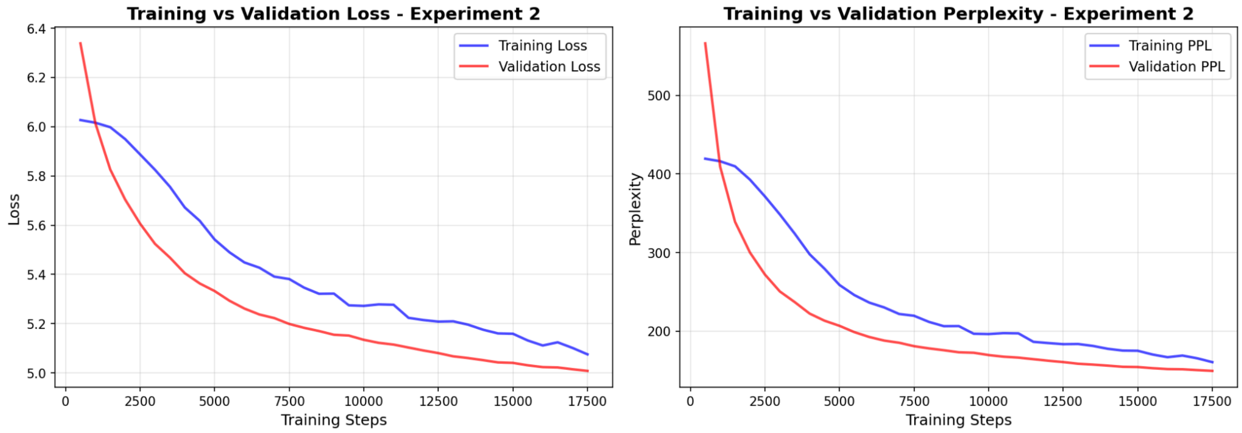

Figure 2: Training and validation loss/perplexity curves for Experiment 2

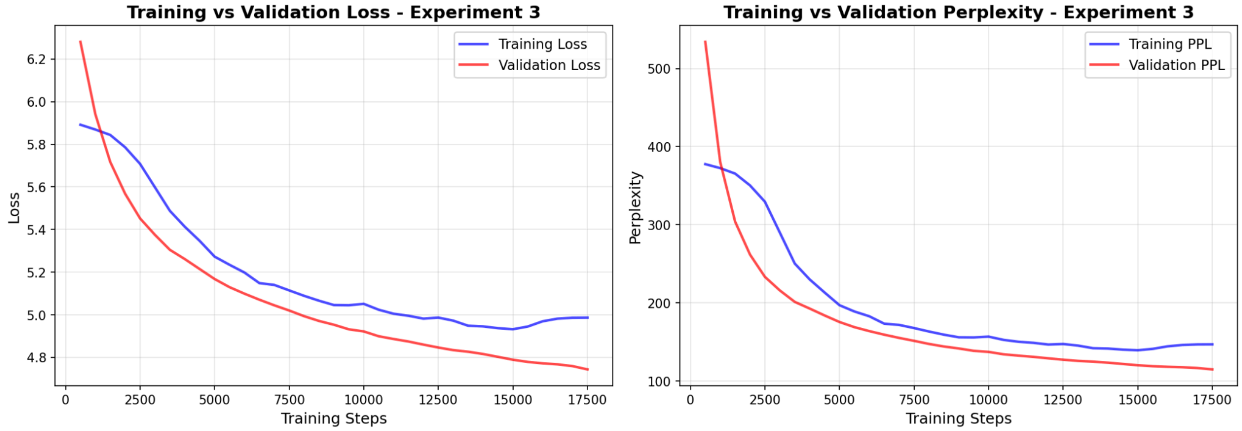

Figure 3: Training and validation loss/perplexity curves for Experiment 3

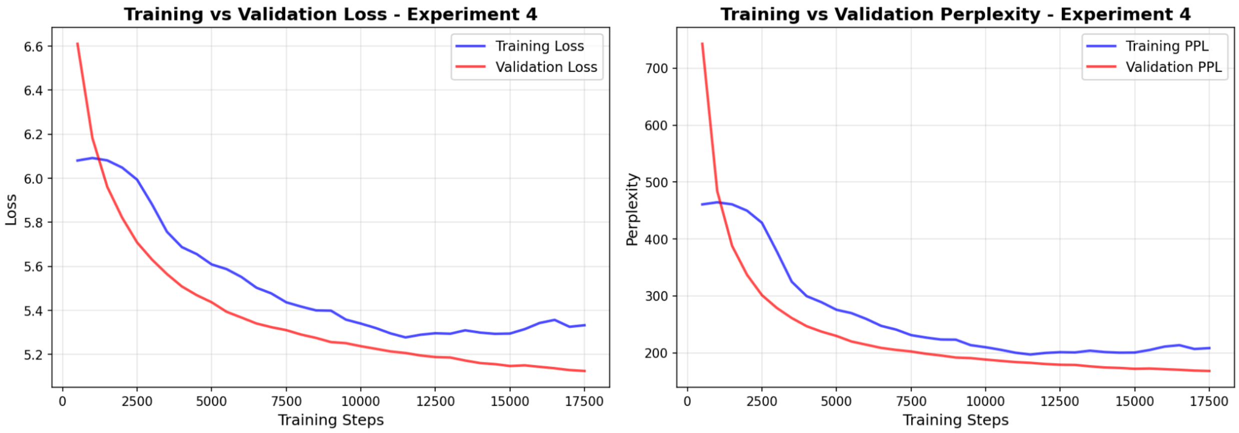

Figure 4: Training and validation loss/perplexity curves for Experiment 4

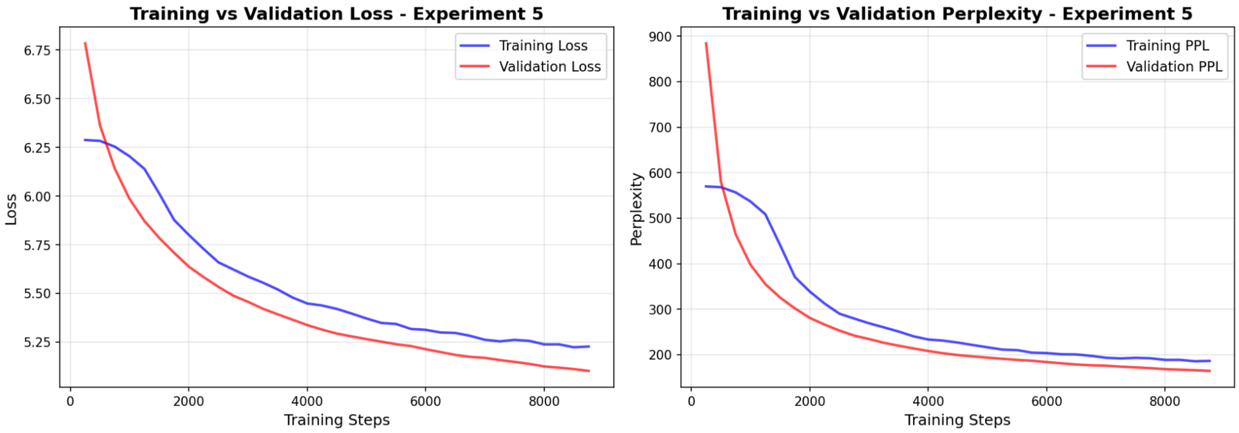

Figure 5: Training and validation loss/perplexity curves for Experiment 5

4.2 Comprehensive Model Comparison

Figure 6 presents a comprehensive comparison of all five experiments, showing training curves, validation curves, and final validation perplexity in a single visualization.

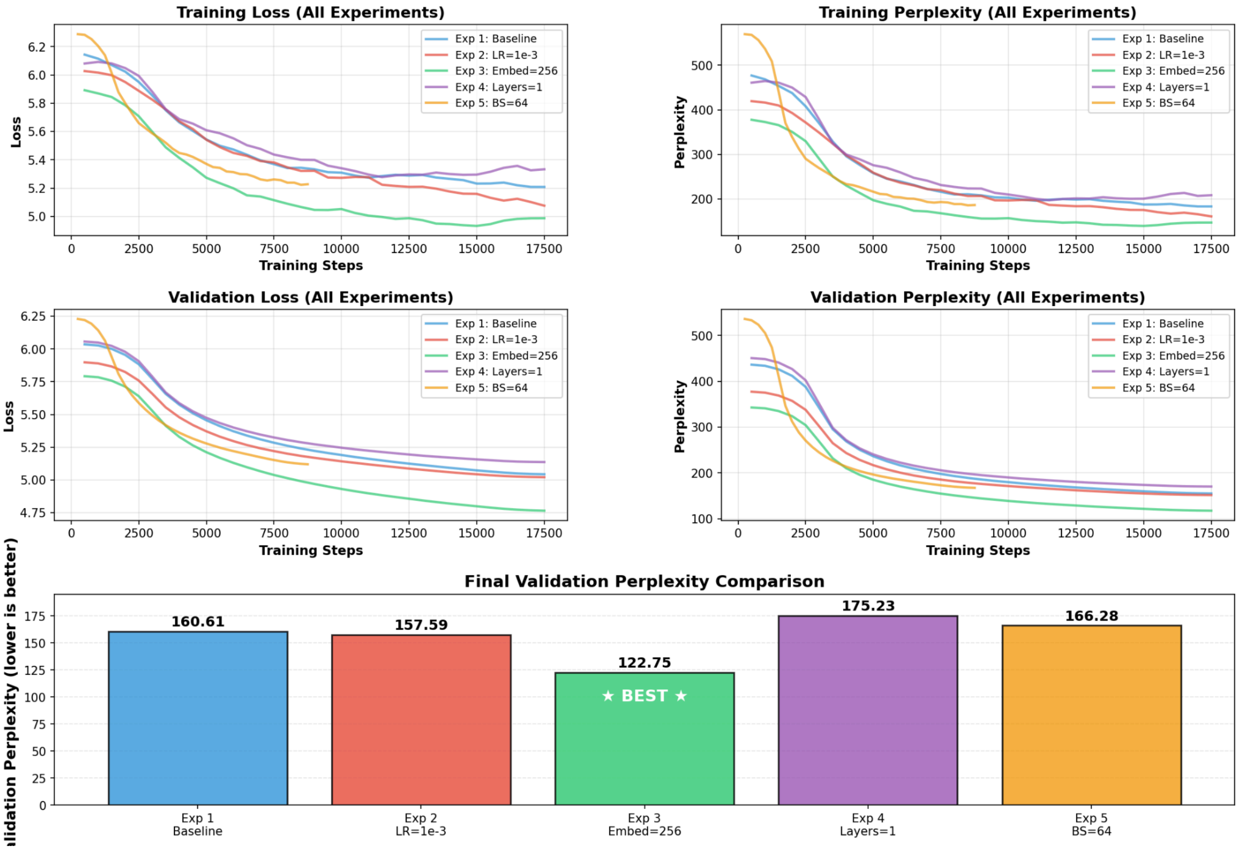

Figure 6: Comprehensive comparison showing training/validation curves and final perplexity rankings across all experiments

4.3 Performance Analysis

- **Best Model:** Experiment 3 (embed=256) achieved the lowest validation perplexity of 122.75, representing a 24% improvement over the baseline (160.61).
- **Model Capacity:** Increasing embedding dimension from 128 to 256 doubled the parameter count (13.3M → 27.3M) and provided the largest performance gain among all experiments.
- **Learning Rate:** Interestingly, higher learning rate (1e-3) performed slightly better than baseline 5e-4 (157.59 vs 160.61), suggesting faster convergence, though still inferior to the larger model.
- **Architecture Depth:** Reducing layers from 2 to 1 resulted in the worst performance (175.23 perplexity), demonstrating that model depth is crucial for learning complex patterns.
- **Batch Size:** Increasing batch size from 32 to 64 degraded performance (166.28 vs 160.61), likely due to fewer gradient updates per epoch (4,396 vs 8,791 steps).

4.4 Training Dynamics

All models showed consistent learning behavior with decreasing training and validation loss over the 2-epoch training period. Key observations include:

- **Convergence:** Smooth loss reduction across all experiments without significant instability

- **No Overfitting:** Validation loss continued to decrease alongside training loss in all experiments
- **Step-Level Monitoring:** 35 checkpoint samples per experiment provided detailed training progression visibility
- **Training Variability:** Training loss exhibited natural batch-level variability, smoothed with moving average (window=10) for visualization clarity, while validation loss was inherently smooth from averaging 10 batches

4.5 Text Generation Examples

The following samples demonstrate the text generation capabilities of the best model (Experiment 3). Text was generated using temperature=0.8 and top-k sampling (k=40) with a maximum of 40 new tokens per prompt.

Prompt 1: "The meaning of life is"
Generated: The meaning of life is something that is supposed to be done in life. the time i realized that i could have found the experience of the age of life has been working at the same time i learned before i felt that i

Prompt 2: "In the year 2050,"
Generated: In the year 2050, by 2050, the growth of this economy may result, the global economy's economy will be higher, and the result of a "fiscal" rate of unemployment in the world's economy. in the

Prompt 3: "Scientists have discovered that"
Generated: Scientists have discovered that theres a new problem. theyre not a thing to do. there are many new types of tools, so i think theyve done it. its important to understand the premise of the process we

Prompt 4: "The president announced"
Generated: The president announced his move to the united states after he was elected president on friday, which he said he was a new one-day peace deal. he said he would get to that house. he said he

Prompt 5: "Once upon a time"
Generated: Once upon a time, he made sure that if you've written it, you'd have to do this, it's not the real thing in the world." "it's the point where you're talking about the time

These samples demonstrate that the model has learned basic language structure and can generate grammatically plausible sentences. The model exhibits reasonable vocabulary usage and maintains some topical coherence with the prompts. However, long-range coherence and factual accuracy remain limited, which is expected for a model of this size (27M parameters) trained on limited data for only 2 epochs.

5. Observations and Challenges

5.1 Key Findings

1. **Model Capacity is Critical:** The embedding dimension had the largest impact on performance. Experiment 3 with 256-dimensional embeddings outperformed all other configurations by a significant margin (24% improvement over baseline).
2. **Learning Rate Effects:** Higher learning rate (1e-3) achieved slightly better results than baseline 5e-4, though both converged successfully. This suggests the model can tolerate a range of learning rates.
3. **Architecture Depth Matters:** Reducing from 2 layers to 1 layer resulted in 9% performance degradation, confirming that even modest depth improvements contribute meaningfully to model capability.
4. **Batch Size Trade-offs:** Larger batch size (64) reduced the number of gradient updates per epoch, leading to worse performance despite more stable gradient estimates.
5. **Train/Val Monitoring:** Proper validation evaluation confirmed no overfitting across all experiments, with validation loss consistently decreasing throughout training.

5.2 Technical Challenges

- **Memory Management:** Larger models (Experiment 3) required more GPU memory and longer training time (~22 min/epoch vs. ~15 min/epoch for baseline).
- **Training Stability:** Gradient clipping (max_norm=1.0) was essential to prevent exploding gradients, especially during early training when loss values were high.
- **Data Loading Efficiency:** With 312K+ samples, efficient data loading with num_workers=2 and pin_memory=True was crucial for maintaining reasonable training times.
- **Checkpoint Persistence:** Saving checkpoints to Google Drive after each epoch prevented data loss from potential runtime disconnections during long training runs.
- **Loss Visualization Strategy:** Training loss required post-hoc smoothing (moving average window=10) to match the inherent smoothness of validation loss, which was averaged over 10 batches during collection.
- **Hyperparameter Space:** Time and computational constraints limited exploration to 5 experiment configurations, though many more combinations could potentially yield further improvements.

5.3 Future Improvements

- **Extended Training:** Training for 5-10 epochs would likely improve performance and allow better assessment of convergence behavior.
- **Learning Rate Scheduling:** Implementing cosine annealing or linear warmup could optimize convergence and potentially improve final performance.
- **Larger Models:** Testing 3-4 layer models or 512 embedding dimensions could yield better results, following the observed trend that capacity improves performance.

- **Regularization Tuning:** The fixed 0.1 dropout rate could be optimized through systematic experimentation.
- **Data Augmentation:** Adding more diverse text sources or implementing data augmentation techniques could improve generalization capabilities.

6. Conclusion

This project successfully implemented a Mini-GPT transformer model and systematically evaluated the impact of key hyperparameters on performance. The best configuration (Experiment 3) achieved a validation perplexity of 122.75 using a 256-dimensional embedding, 2 transformer layers, 5e-4 learning rate, and batch size of 32.

The experiments demonstrated that model capacity, measured through embedding dimension and layer count, has the most significant impact on performance. Increasing embedding dimension from 128 to 256 resulted in a 24% improvement in validation perplexity, despite doubling the parameter count and training time. Learning rate and batch size showed more modest effects, with the baseline values proving near-optimal for this model and dataset.

The model exhibits reasonable text generation capabilities for its size, producing grammatically correct sentences with some topical coherence. While long-range coherence and factual accuracy remain limited (as expected for a 27M parameter model trained on limited data), the generated text demonstrates that the model has successfully learned basic language patterns and structure.

This hands-on implementation provided valuable insights into transformer architecture, training dynamics, and the practical considerations of building language models from scratch. The systematic hyperparameter experiments revealed clear performance patterns: model capacity matters most, followed by architecture depth, with learning rate and batch size having smaller effects. These findings align with established principles in the deep learning literature and provide a solid foundation for understanding modern large language models.

7. References and Resources

- **Model Architecture:** Based on GPT architecture (Radford et al., 2018)
- **Implementation Framework:** PyTorch 2.9.0
- **Tokenizer:** GPT-2 BPE from Hugging Face Transformers
- **Training Environment:** Google Colab with NVIDIA T4 GPU
- **Dataset Sources:** WikiText-103, OpenWebText (preprocessed in Assignment 1)
- **Data Processing:** Hugging Face datasets library for data loading and streaming