Assignment 2: Neural Language Model Training (PyTorch)

Author

Satwik Rakhelkar
Assignment for IIIT Hyderabad Internship

Objective

The goal of this assignment is to implement a character-level neural language model from scratch using PyTorch. The model is trained to predict the next character in a sequence, using *Pride and Prejudice* by Jane Austen as the dataset. The assignment explores how model capacity and training duration affect learning behavior, and evaluates performance using loss curves and perplexity.

Dataset

The dataset used is the full text of *Pride and Prejudice* by Jane Austen. It is processed using character-level tokenization. All preprocessing steps including reading the file, lowercasing, vocabulary creation, encoding, and batching are implemented manually using Python and PyTorch.

Model Architecture

The model is a simple character-level LSTM implemented using PyTorch's nn.Module. It consists of:

- An embedding layer to convert character indices into dense vectors
- An LSTM layer to model sequential dependencies
- A linear layer to project LSTM outputs to vocabulary logits

The model is trained using the Adam optimizer and CrossEntropyLoss. All training is performed on GPU (Colab environment).

Note: I used Microsoft Copilot to help understand the assignment structure and refine parts of the code.

Experimental Setup

Three experiments were conducted to demonstrate underfitting, overfitting, and best-fit behavior. Each experiment varies the model size and training duration.

1. Underfit

- **Embedding Dimension**: 32
- **Hidden Dimension**: 64
- **Epochs**: 5
- **Dataset**: Full text
- **Perplexity**: 3.56

Observation: The model is too small and trained for too few epochs to capture meaningful patterns. Loss decreases slightly but plateaus early, indicating underfitting.

2. Overfit

- **Embedding Dimension**: 256
- **Hidden Dimension**: 512
- **Epochs**: 30
- **Dataset**: First 10,000 characters
- **Perplexity**: 1.05

Observation: The model memorizes the small dataset. Loss drops rapidly and stabilizes at a very low value. This confirms overfitting, as the model performs well on training data but would generalize poorly.

3. Best Fit

- **Embedding Dimension**: 128
- **Hidden Dimension**: 256
- **Epochs**: 15
- **Dataset**: Full text
- **Perplexity**: 3.35

Observation: The model shows balanced learning. Loss decreases steadily until around epoch 8, after which it begins to rise again, indicating mild overtraining. This configuration offers the best trade-off between capacity and generalization.

<u>Results Summary</u>

Three experiments were conducted to demonstrate underfitting, overfitting, and best-fit behavior. In the underfit scenario, the model was configured with an embedding dimension of 32 and a hidden dimension of 64, trained for 5 epochs on the full dataset. It achieved a perplexity of 3.56, indicating minimal learning due to limited capacity and training time.

In the overfit scenario, the model used a much larger configuration: embedding dimension of 256 and hidden dimension of 512, trained for 30 epochs on a small subset of the dataset (first 10,000 characters). It achieved a very low perplexity of 1.05, showing that it had memorized the training data and would likely generalize poorly.

The best fit configuration used an embedding dimension of 128 and a hidden dimension of 256, trained for 15 epochs on the full dataset. It achieved a perplexity of 3.35, with a loss curve that decreased steadily until around epoch 8, after which it began to rise again. This suggests mild overtraining and indicates that early stopping could improve generalization.

Loss plot for each experiment is included below to visually support these observations.

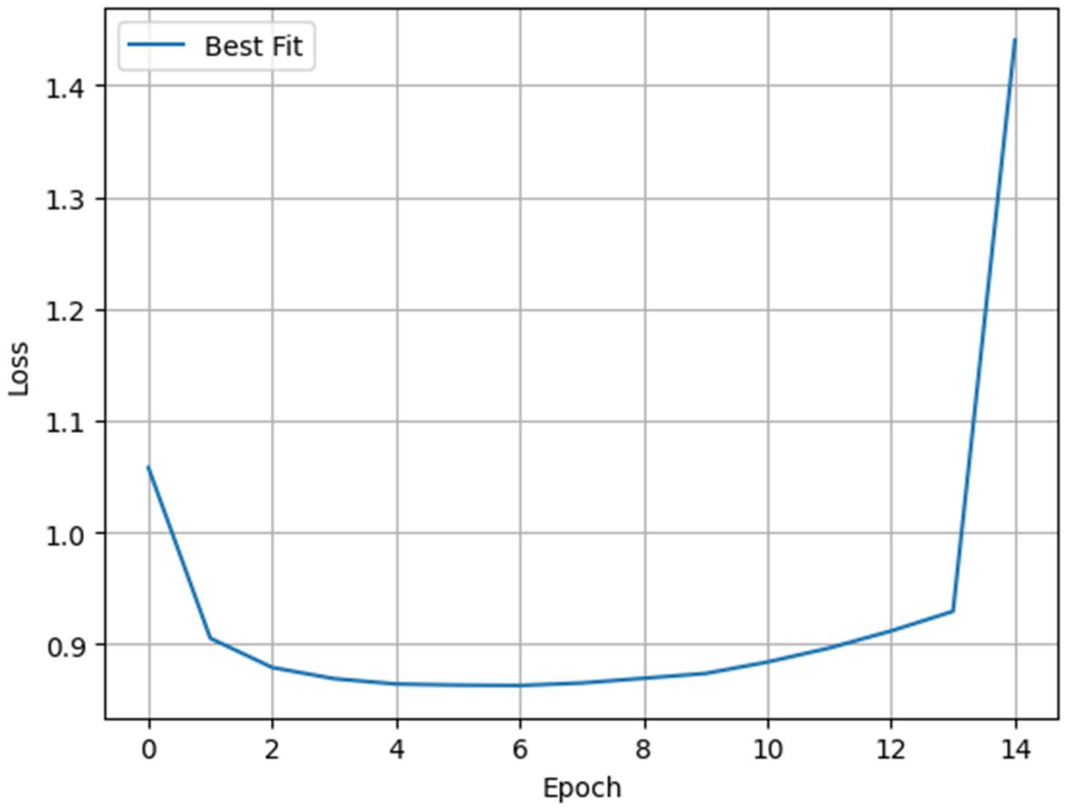

Fig1: Best fit Loss Curve

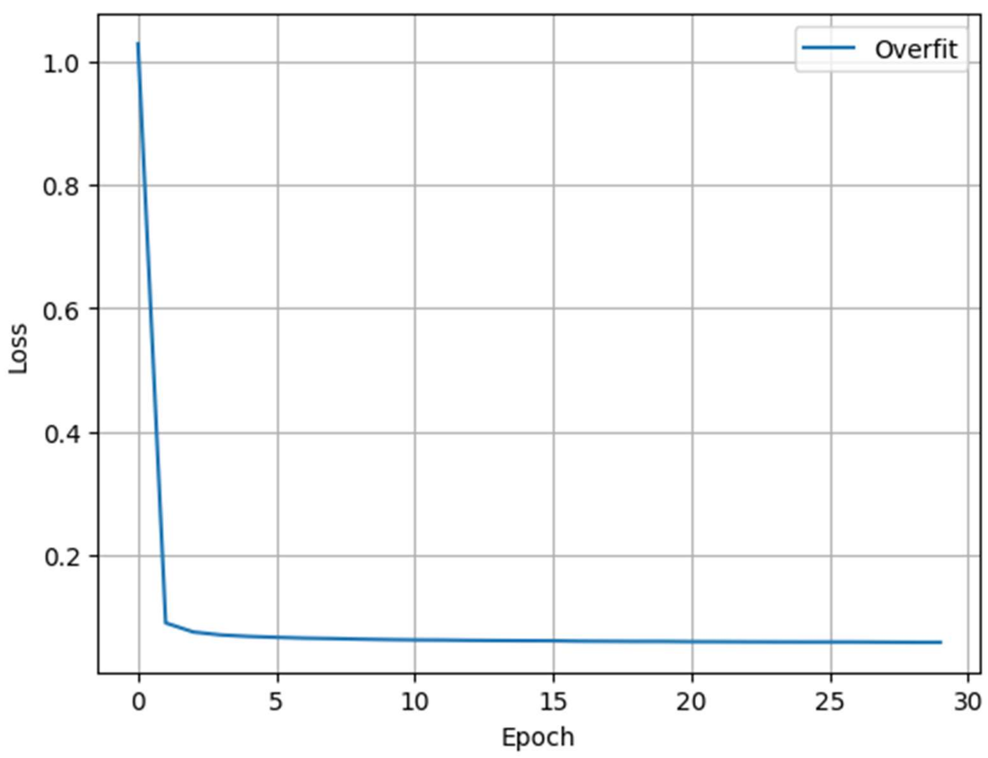

Fig2: Overfit Loss Curve

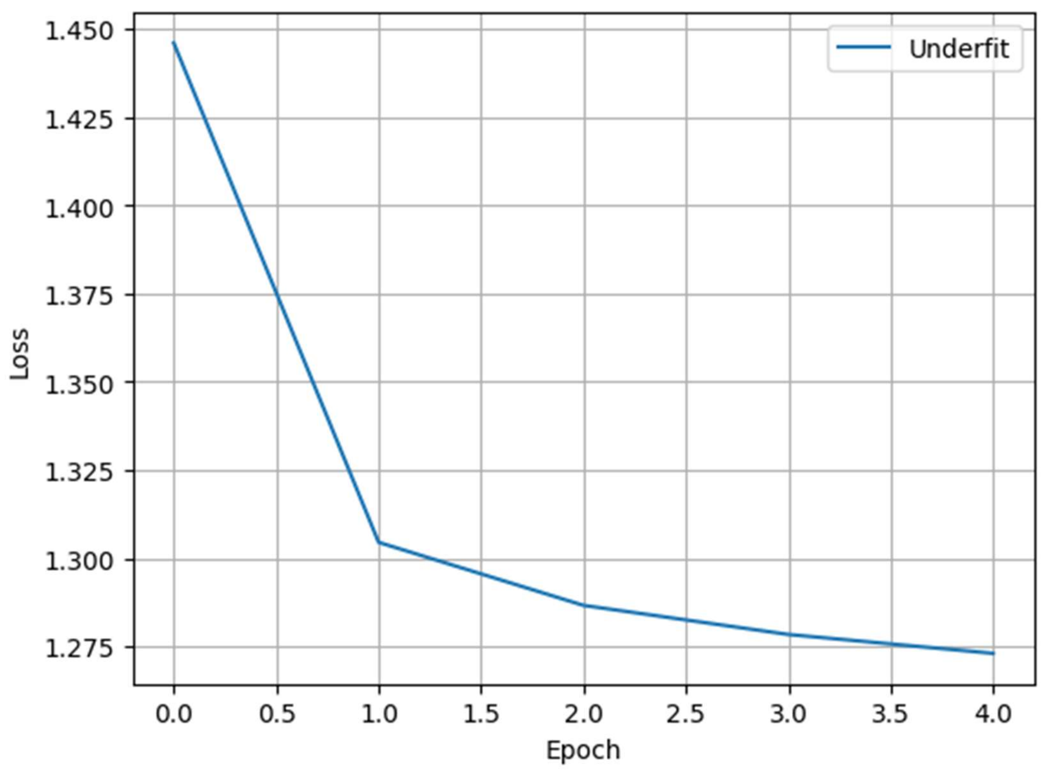

Fig3: Underfit Loss Curve

Reproducibility

- Random seed is fixed (seed=42) for consistent results
- All code is implemented from scratch using PyTorch
- No external libraries or pre-trained models were used
- Training and evaluation are performed in Google Colab with GPU support

Files Submitted

- Assignment2.ipynb: Full training and evaluation notebook
- report.pdf: This report
- loss_plot_underfit.png: Underfit loss curve
- loss_plot_overfit.png: Overfit loss curve
- loss_plot_bestfit.png: Best fit loss curve
- README.md: GitHub instructions and summary

Conclusion

This assignment demonstrates how model size and training duration affect learning behavior in neural language models. Through three controlled experiments, the model's ability to generalize, memorize, or underfit is clearly observed. The best-fit configuration achieves a good balance between capacity and generalization, with a perplexity of 3.35 on the full dataset.