Assignment 2 Report: Neural Language Model Training (PyTorch)

Name **: Barla Sowmya** Date:14th Nov 2025

1. Objective

The goal of this assignment was to train a neural language model from scratch using PyTorch. The main objectives were:

1. To understand how sequence models (RNN, GRU, LSTM) predict text.
2. To examine the effect of model design, capacity, and training duration on performance.
3. To implement experiments to demonstrate underfitting, best-fit, and overfitting.
4. To evaluate model performance using training/validation loss and perplexity.

2. Dataset

- The dataset provided consisted of -Pride_and_Prejudice-Jane_Austen.txt
- Link https://drive.google.com/file/d/1FNuJSYRdN2BomupmNkkKROBSkQzSPS9N/view?usp=sharing
- **Preprocessing steps:**
    1. Converted all text to lowercase.
    2. Tokenized at **character level**.
    3. Created a vocabulary of unique characters.
    4. Mapped characters to integer indices (char2idx) and vice versa (idx2char).
    5. Encoded the full text as integer sequences.
    6. Prepared input sequences of length SEQ_LEN=50 and corresponding targets (next character).
    7. Split data into **90% training** and **10% validation** sets.

3. Model Implementation

- Chosen a **GRU (Gated Recurrent Unit)** architecture for sequence modeling due to its efficiency and ability to capture long-term dependencies as Transformers need large amounts of Data, RNN's performance

Why I chose GRU:

- GRUs are efficient and require fewer parameters than LSTMs, allowing faster training while still effectively capturing long-term dependencies in sequences.
- Unlike basic RNNs, GRUs solve the vanishing gradient problem, making them much better at learning from long texts and remembering context over many time steps.
- On moderate-size datasets like this assignment, GRUs typically offer a good balance between model complexity and performance, providing accuracy close to LSTMs but with simpler architecture and less computational cost.

Why not vanilla RNNs or LSTMs/Transformers:

- **Vanilla RNNs:** Suffer from vanishing/exploding gradients and quickly forget information, leading to poor performance on language modeling tasks involving long input sequences.
- **LSTM:** Provides excellent long-term memory, but is more complex (needs more gates, has more parameters), resulting in longer training time for similar performance, especially on medium or small datasets.
- **Transformers:** Very powerful but require much larger datasets and computational resources to outperform GRUs/LSTMs on classic text tasks. For smaller datasets, they can overfit or underperform compared to compact GRU models.

Architecture of GRU LM:

1. **Embedding layer:** Converts character indices to dense vectors (embed_dim).
2. **GRU layer:** Captures sequential dependencies (hidden_dim, num_layers).
3. **Fully connected layer:** Maps hidden states to vocabulary logits for next-character prediction.

- **Hyperparameters varied** across experiments: embedding size, hidden size, number of layers, dropout, learning rate, and training epochs.

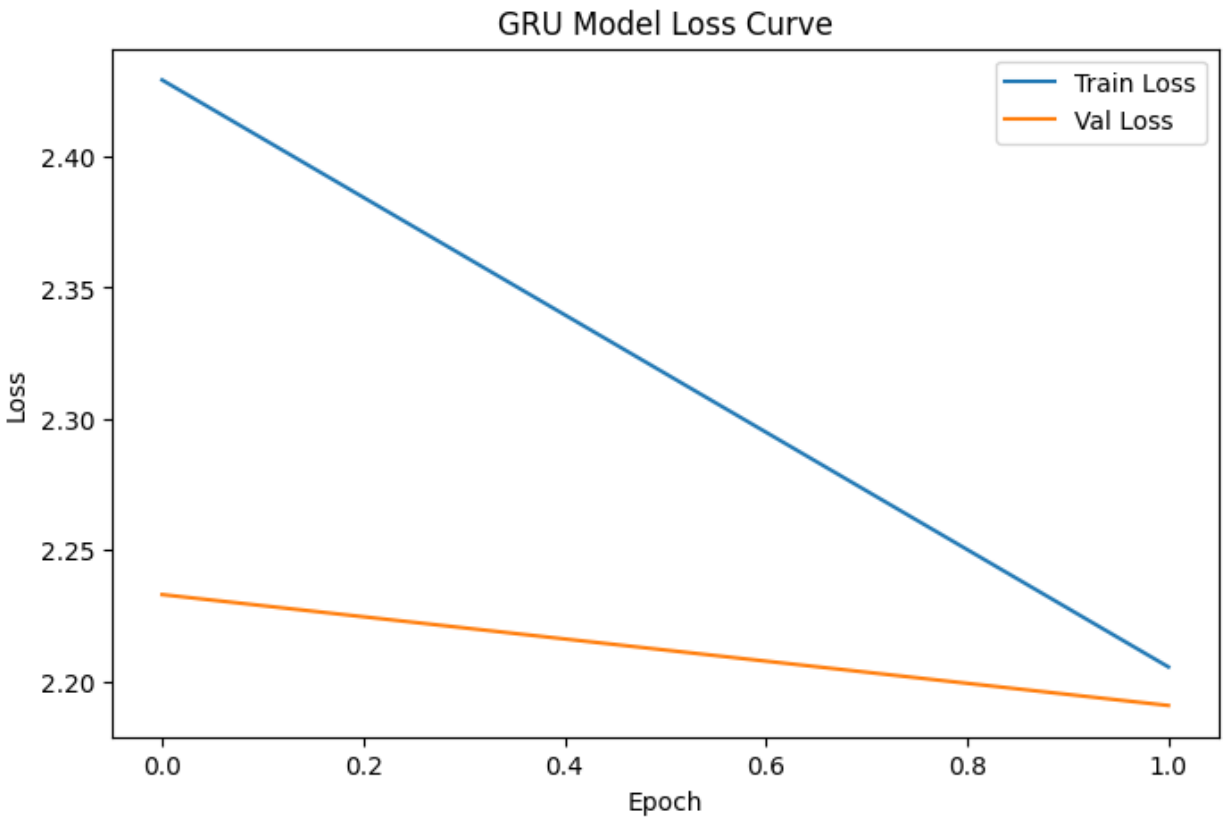
Hyperparameter Tuning:

Tried different hyperparameters and selected values based on validation loss,perplexity and learning curves.

4. Experiments

4.1 Underfitting Scenario

- **Configurations:**
  - Embedding size =8
  - hidden_dim=8,
  - num_layers=1
  - sequence length = 50
  - Batch size =64
  - number of training epochs: 2
  - patience =2

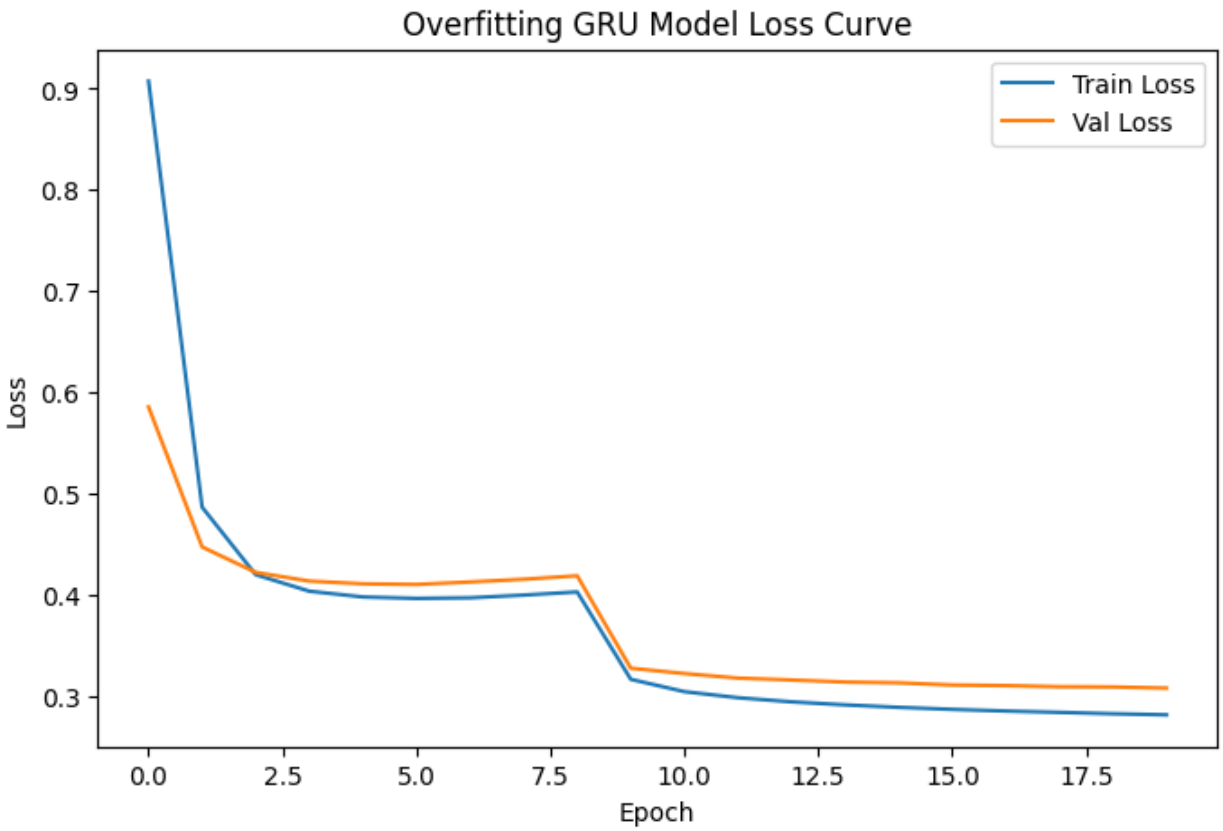


- **Observations:**
  - Training loss remained high.
  - Validation loss also high and almost flat.
  - Generated text was mostly random, no meaningful sequences.
- **Conclusion:**
  Low model capacity and insufficient training caused **underfitting**. The model failed to learn even training patterns.

4.2 Overfitting Scenario

- **Configurations:**
  - Embedding size =256
  - hidden_dim=512
  - num_layers=3

- sequence length = 50
- Batch size =64
- number of training epochs: 20
- dropout = 0.1

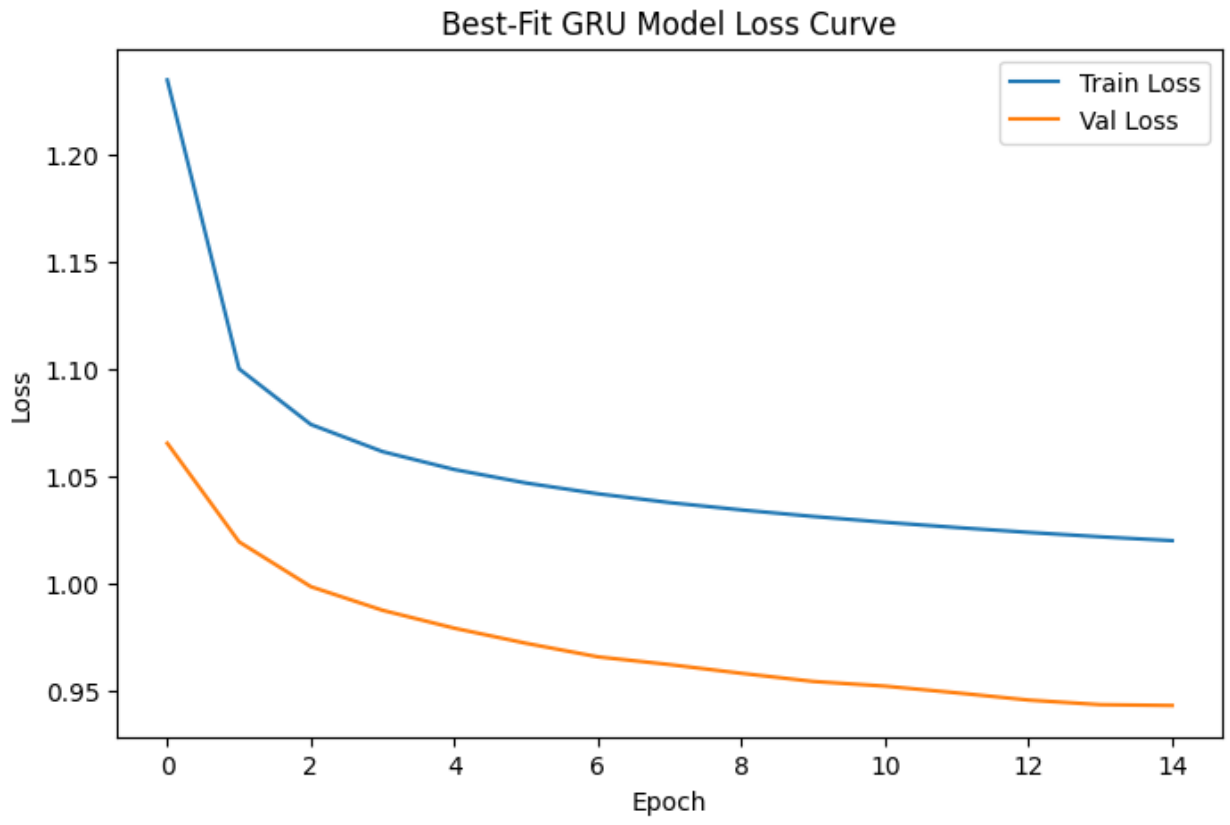


- **Observations:**
  - The train loss is lower than validation loss and continues to decrease even after the validation loss flattens.
  - Validation loss decreased initially but then increased → mild **overfitting**.
  - There is a noticeable gap between train and validation loss at later epochs.

- **Conclusion:**
  Large capacity with long training leads to overfitting. The model memorizes training data without generalizing.

4.3 Best-Fit Scenario

- **Configurations:**
    - Embedding size =128
    - hidden_dim=256,
    - num_layers=2
    - sequence length = 50
    - Batch size =64
    - number of training epochs: 15
    - dropout = 0.3
    - **Early stopping** applied (patience=3) to prevent overfitting.



- **Observations:**
    - Both train and validation loss decrease steadily
    - The gap between train and val loss is small and consistent.

- The curve spans ~15 epochs, giving the model time to converge.

- **Conclusion:**
  Balanced model capacity and proper training produced the **best-fit model**, learning patterns without memorization.

Scenario	Epochs	Embed Dim	Hidden Dim	Layers	Train Loss	Val Loss	Observations
Underfit	5	32	64	1	2.1889	2.1909	Cannot learn patterns
Best-Fit	15-20	128	256	2	0.2818	0.3082	Balanced, optimal
Overfit	30	256	512	3	0.2818	0.3082	Memorizes training

Loss Curves:

- Underfit

  Train Loss: 2.1889 | Train Perplexity: 8.9250

  Val Loss: 2.1909 | Val Perplexity: 8.9431

- Best-fit:

  Train Loss: 0.9358 | Train Perplexity: 2.5493

  Val Loss: 0.9431 | Val Perplexity: 2.5680

- Overfit:

  Train Loss: 0.2818 | Train Perplexity: 1.33

  Val Loss: 0.3082 | Val Perplexity: 1.36

6. Evaluation Metric

- **Perplexity:**
  - Perplexity = exp(validation loss)
  - Lower perplexity indicates better model predictions.

- Observed:
  - Underfit: High perplexity
  - Best-fit: Moderate, optimal perplexity
  - Overfit: Low training perplexity, high validation perplexity

7. Key Learnings

1. **Model capacity affects performance**:
   - Too small → underfitting
   - Too large → overfitting
   - Balanced → best-fit
2. **Training duration matters**:
   - Too short → underfitting
   - Too long without early stopping → overfitting
3. **Early stopping** prevents overfitting in large models.
4. **Observing training vs validation loss** is critical to diagnose fit.
5. **Text generation** helps qualitatively check learning.
6. **Reproducibility** via random seeds ensures consistent results.

Rationale for selecting the Best fit motel:

- The best model has the lowest validation loss and perplexity, meaning it generalizes best and predicts unseen data more accurately.
- Its training loss decreases steadily without a huge gap from validation loss, showing it learns well but does not memorize (overfit) the training set.
- The loss curves are stable with no sudden spikes, and text generations from the model are fluent and contextually appropriate, confirming effectiveness beyond just numeric metrics.

8. Conclusion

Through these experiments, I learned the **impact of model design, training strategy, and hyperparameters** on language model performance. The best-fit GRU model balanced capacity and training, while underfitting and overfitting demonstrated the consequences of insufficient or excessive learning. Loss curves, perplexity, and generated text were essential to evaluate the model's understanding and generalization.

**Full code and reproducibility instructions are provided in the attached GitHub repository.

Link :

https://github.com/BarlaSowmya/-Neural-Language-Model-Training-PyTorch