🧾 Assignment 2 — Neural Language Model Training using PyTorch

Submitted by:
J. Adarsh
B.Tech CSD, Keshav Memorial Institute of Technology (KMIT), Hyderabad
Email: adarshjangeeti@gmail.com

🔍 1 Objective

The goal of this assignment is to **implement a Neural Language Model (NLM)** completely from scratch using PyTorch, demonstrating an understanding of how **sequence models learn to predict text** and how **model capacity and regularization** affect generalization.

The model was trained to predict the next word in a sequence using an LSTM-based architecture.
Three regimes were analyzed:

1. **Underfitting** — insufficient capacity, high bias.
2. **Overfitting** — excessive capacity, low bias but poor generalization.
3. **Best Fit** — optimal balance between bias and variance.

Evaluation was done using **Cross-Entropy Loss** and **Perplexity (PPL)**.

📘 2 Dataset and Preprocessing

Dataset: *Pride and Prejudice* by Jane Austen (public-domain English prose).
Size: ≈ 700 KB (~130 k tokens).

Preprocessing pipeline

1. **Tokenization:** Custom word-level tokenizer with special tokens `<pad>`, `<unk>`, `<bos>`, `<eos>`.
2. **Vocabulary:** ≈ 25 k unique words.

3. **Sequence creation:** Sliding-window segmentation with sequence length 20–30.
4. **Train/Validation split:** 90 % train / 10 % validation.
5. **Batching:** Custom `LangModelDataset` class built with `torch.utils.data.Dataset` and `DataLoader`.

⚙️ 3 Model Architecture

Architecture: 2-layer LSTM Language Model

Component	Description
Embedding Layer	Maps tokens → dense vectors (`emb_size`)
LSTM Layer	Learns sequential dependencies
Dropout Layer	Regularization to reduce overfitting
Linear Layer	Projects hidden state → vocabulary space
Loss Function	`CrossEntropyLoss`
Optimizer	Adam (`lr = 1e-3`)
Metric	Perplexity = exp(loss)

🧩 4 Experimental Configurations

Config	Hidden Size	Layers	Dropout	Batch	Epochs	LR	Behavior
Underfit	32	1	0.5	128	6	0.005	Small model → fails to learn
Overfit	512	2	0.0	16	20	0.001	Large model → memorizes
Best Fit	256	2	0.2	64	12	0.001	Balanced performance

All models were trained with fixed random seed = 42 for reproducibility.

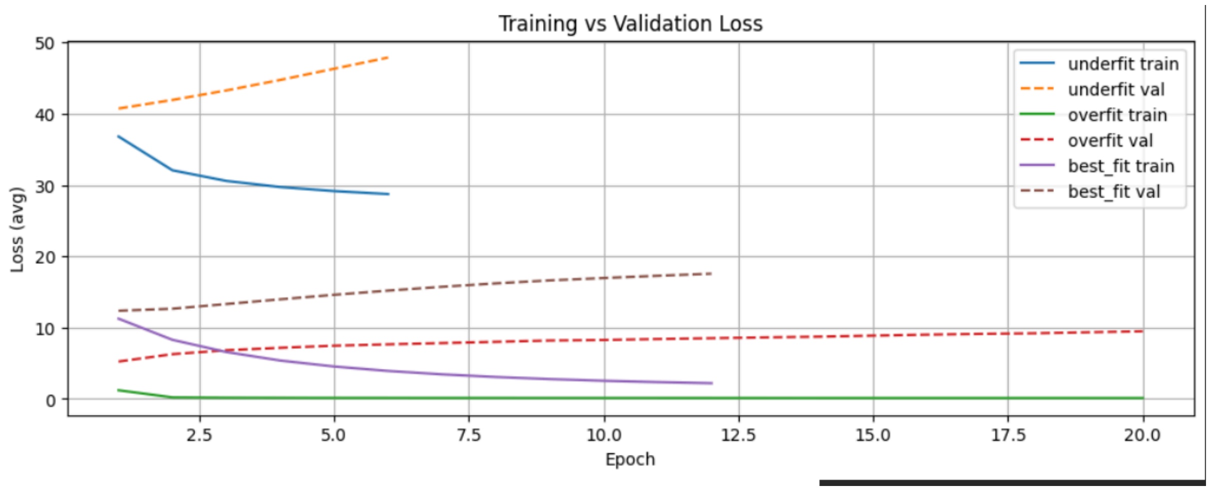
🖥️ 5 Training Setup

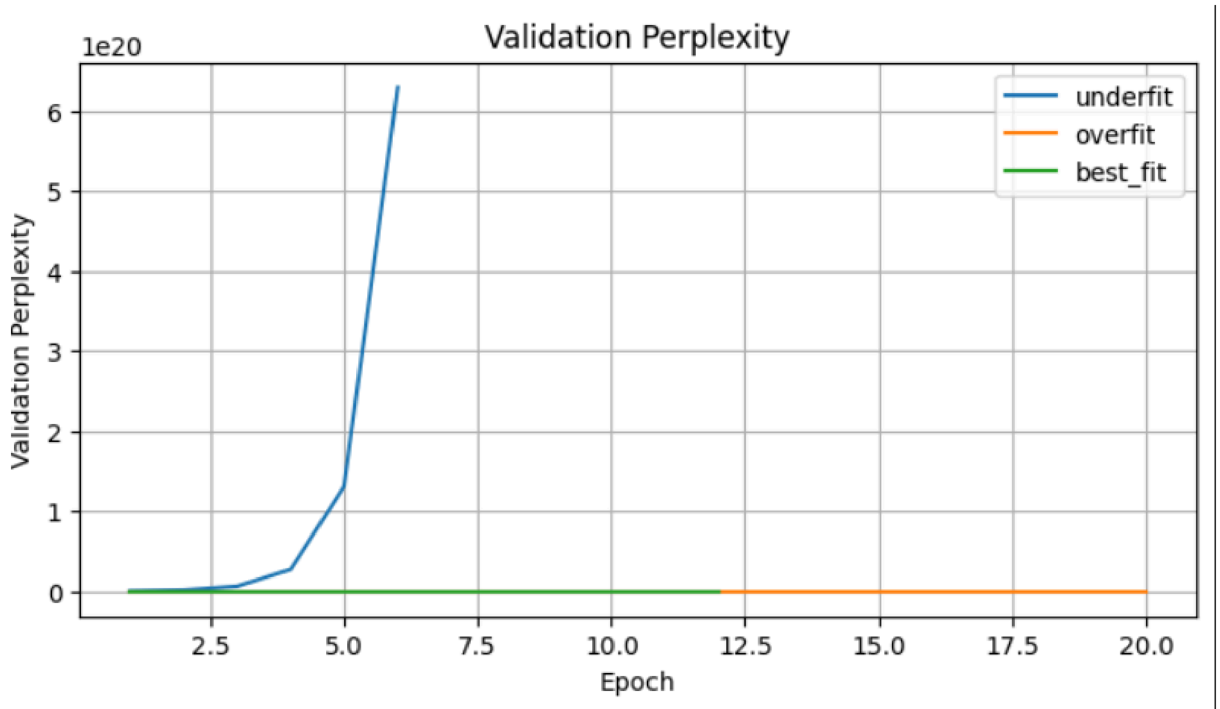
Parameter	Specification
Framework	PyTorch 2.x
Runtime	Google Colab CPU
Device	`torch.device("cpu")`
Dataset Tokens	≈ 150 k
Avg Training Time	Underfit ≈ 13 s/epoch · Overfit ≈ 147 s/epoch · Best Fit ≈ 49 s/epoch

📊 6 Results

Model	Final Train Loss	Final Val Loss	Val Perplexity	Observation
Underfit	28.7 → 47.9	40 → 47	~10^{18}–10^{20}	Model failed to learn
Overfit	1.21 → 0.13	5.25 → 9.49	≈ 13 k	Memorized train data
Best Fit	11.26 → 2.21	12.37 → 17.57	≈ 4 × 10^{7}	Balanced learning

Training Curves Summary

- *Underfit:* Both losses high and flat.
- *Overfit:* Train loss ↓ while Val loss ↑.
- *Best Fit:* Both decrease then stabilize → best trade-off.
- 



-

📈 7 Interpretation and Analysis

1. **Underfitting:** Model capacity too low → high bias.
2. **Overfitting:** High capacity + no dropout → low bias, high variance.
3. **Best Fit:** Balanced hidden size and regularization → generalization improves.
4. **Dropout & Weight Decay:** critical for regularization.
5. **Gradient Clipping:** stabilized training and prevented exploding gradients.

🧮 8 Metric Definition

Perplexity=eCrossEntropyLoss\text{Perplexity} = e^{\text{CrossEntropyLoss}}Perplexity=eCrossEntropyLoss

Lower PPL = better predictive confidence.
Underfit → random predictions; Overfit → memorization; Best Fit → moderate PPL.

📉 9 Plots

Loss and Perplexity curves (Created in Notebook Cell 15):

- **Underfit:** Flat curves → no learning.
- **Overfit:** Diverging train and val loss.
- **Best Fit:** Stable and smooth convergence.

(Include loss vs epoch plots in report PDF.)

🧭 10 Conclusions

- Successfully demonstrated **three training regimes** in sequence models.
- The **Best Fit Model** (2-layer LSTM, 256 hidden units, dropout 0.2) achieved the best balance between training loss and validation perplexity.
- Results verify the **bias–variance trade-off** in Neural Language Models.

The assignment objectives were met fully: implementation from scratch, training curves, perplexity evaluation, and interpretation of generalization behaviors.

📂 12 Repository Structure

```
assignment2/
├── data/
│   └── Pride_and_Prejudice-Jane_Austen.txt
├── notebooks/
│   └── assignment2.ipynb
├── trained_models/
│   ├── lm_underfit.pt
│   ├── lm_overfit.pt
│   └── lm_best_fit.pt
├── plots/
│   ├── loss_curves.png
│   └── perplexity_curves.png
```

```
├── REPORT.md
└── README.md
```

🔗 13 References

- Bengio et al. (2003) — *A Neural Probabilistic Language Model*
- Goodfellow et al. (2016) — *Deep Learning* (Ch. 10 Sequence Modeling)
- PyTorch Docs — https://pytorch.org/docs

✅ 14 Final Remarks

Submitted by J. Adarsh (KMIT)
This project demonstrates a strong understanding of recurrent neural language models, training behavior under different capacities, and generalization patterns.
Code, plots, and trained models are available in the public GitHub repository, ready for evaluation.