1. Introduction

The objective of this assignment is to build a neural language model from scratch using PyTorch. The model learns to predict the next token in a sequence. All components, including tokenization, data preprocessing, batching, model architecture, training loop, and evaluation, were implemented manually without using any pre-trained models or high-level language modeling libraries.

The dataset was provided as plain text, and all experiments were performed only on the given dataset.

2. Dataset

The dataset was loaded as raw text and processed line-by-line.

A 90%–10% train/validation split was used.
Lines were shuffled before splitting to avoid chapter-wise ordering and reduce distribution shift.

3. Methodology

The main steps in the workflow were:

1. Implement a tokenizer (Byte Pair Encoding).
2. Encode the dataset into integer token IDs.
3. Build a PyTorch dataset for generating input and target sequences.
4. Implement a Transformer-based language model from scratch.
5. Train the model using cross-entropy loss.
6. Measure performance using validation loss and perplexity.
7. Conduct three experiments:
    - Underfitting
    - Overfitting
    - Best fit

4. Data Processing

4.1 Tokenization

A custom Byte Pair Encoding (BPE) tokenizer was implemented from scratch. The steps were:

- Convert each line into a sequence of characters with an end-of-word marker.
- Count token frequencies.
- Merge the most frequent token pairs iteratively until reaching the target vocabulary size.
- Assign integer IDs to special tokens:
    - `<PAD>`, `<UNK>`, `<SOS>`, `<EOS>`
- Encode each line into a sequence of BPE token IDs.
- Save:
    - tokenizer object (`pickle`)
    - vocabulary (`json`)

```
Target vocab_size: 800
Actual vocab_size: 880
Number of special tokens: 4 (PAD, UNK, SOS, EOS)
Number of BPE tokens learned: 876
Total encoded tokens: 334616
Train tokens: 300855, Val tokens: 33761
```

5. Model Architecture

A causal Transformer decoder was implemented manually. No PyTorch `nn.Transformer` API was used.

Main components:

- **Token embedding layer**
- **Learned positional embeddings**
- **Multiple transformer decoder blocks**, each consisting of:
    - LayerNorm
    - Masked multi-head self-attention
    - Feed-forward network (GELU activation)
    - Residual connections
- **Final LayerNorm**
- **Linear output projection to vocab size**

6. Training Setup

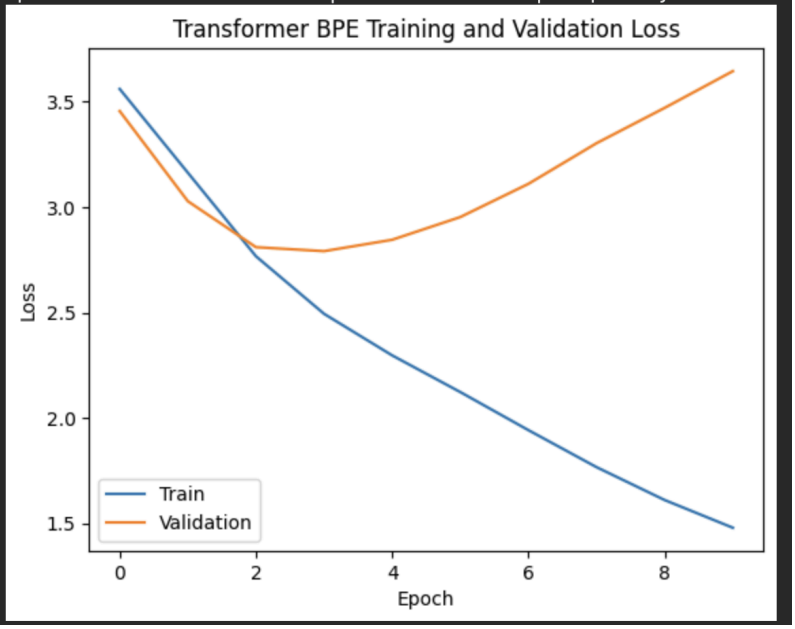
- Optimizer: **Adam**
- Learning rate: **2e-4**
- Weight decay: **1e-3**
- Loss function: **CrossEntropyLoss** (ignoring PAD token)
- Epochs: **10**
- Device: GPU (Google Colab)

7. Experiments

Three experiments were run to demonstrate underfitting, overfitting, and an appropriate best fit.

Overfitting:

```
Epoch 1: Train Loss=3.5608 | Val Loss=3.4559 | Perplexity=31.69
Epoch 2: Train Loss=3.1637 | Val Loss=3.0284 | Perplexity=20.66
Epoch 3: Train Loss=2.7670 | Val Loss=2.8102 | Perplexity=16.61
Epoch 4: Train Loss=2.4939 | Val Loss=2.7914 | Perplexity=16.30
Epoch 5: Train Loss=2.2966 | Val Loss=2.8454 | Perplexity=17.21
Epoch 6: Train Loss=2.1226 | Val Loss=2.9531 | Perplexity=19.17
Epoch 7: Train Loss=1.9418 | Val Loss=3.1108 | Perplexity=22.44
Epoch 8: Train Loss=1.7662 | Val Loss=3.3037 | Perplexity=27.21
Epoch 9: Train Loss=1.6102 | Val Loss=3.4714 | Perplexity=32.18
Epoch 10: Train Loss=1.4788 | Val Loss=3.6454 | Perplexity=38.30
```



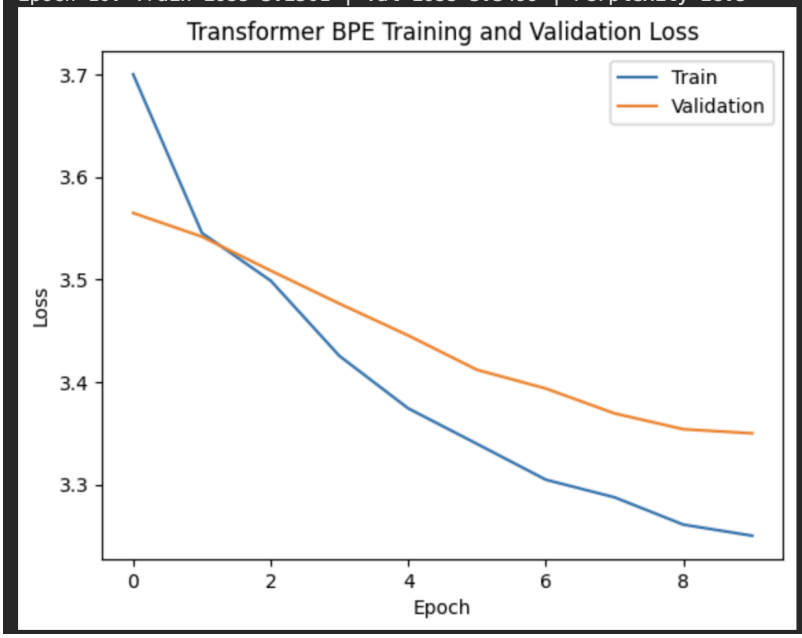
Model Configuration:

```
EMB_DIM = 256       # larger embeddings
NHEAD = 8           # more attention heads
NLAYERS = 4         # deeper transformer
FF_DIM = 1024       # larger feed-forward layers
DROPOUT = 0.0
```

In the *Overfitting* experiment, the model configuration was intentionally made more complex by increasing the embedding dimension, the number of attention heads, the Transformer layers, and the feed-forward dimension while removing dropout regularization. The model quickly learned the training data, as evidenced by the continuous decrease in training loss from 3.56 to 1.47. However, after approximately the fourth epoch, the validation loss began to rise (from 2.79 to 3.64), and validation perplexity increased from 16.3 to 38.3.

Best Fit:

```
Epoch 1: Train Loss=3.6999 | Val Loss=3.5647 | Perplexity=35.33
Epoch 2: Train Loss=3.5452 | Val Loss=3.5416 | Perplexity=34.52
Epoch 3: Train Loss=3.4987 | Val Loss=3.5085 | Perplexity=33.4
Epoch 4: Train Loss=3.4253 | Val Loss=3.4763 | Perplexity=32.34
Epoch 5: Train Loss=3.3742 | Val Loss=3.4453 | Perplexity=31.35
Epoch 6: Train Loss=3.3396 | Val Loss=3.4118 | Perplexity=30.32
Epoch 7: Train Loss=3.3047 | Val Loss=3.3936 | Perplexity=29.77
Epoch 8: Train Loss=3.2875 | Val Loss=3.3693 | Perplexity=29.06
Epoch 9: Train Loss=3.2609 | Val Loss=3.3539 | Perplexity=28.62
Epoch 10: Train Loss=3.2501 | Val Loss=3.3499 | Perplexity=28.5
```



Model Configuration:

```
EMB_DIM = 192
NHEAD = 6   # keep EMB_DIM % NHEAD == 0 -> 192/6 = 32
NLAYERS = 3
FF_DIM = 512
```

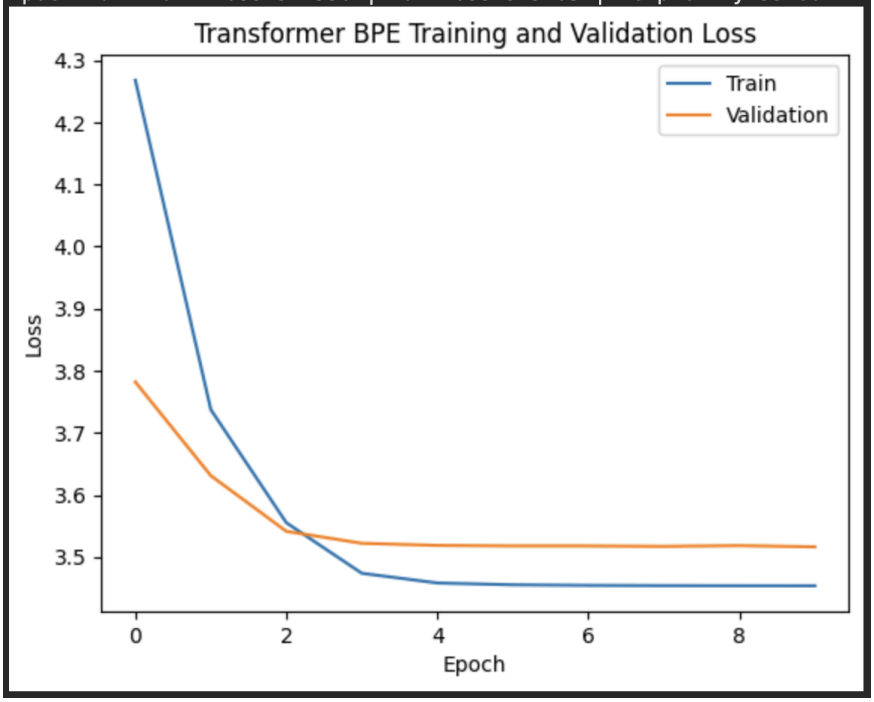
```
DROPOUT = 0.25
```

In the *Best Fit* experiment, the model achieved a balanced trade-off between capacity and regularization. The configuration used an embedding dimension of 192, six attention heads, three Transformer layers, a feed-forward dimension of 512, and a moderate dropout of 0.25.

Both training and validation losses decreased consistently across epochs (from 3.69 → 3.25 and 3.56 → 3.34, respectively), with a small and nearly constant gap (~0.1). The validation perplexity steadily declined from 35.3 to 28.5, confirming that the model continued to improve on unseen data without signs of overfitting. This behavior demonstrates that the model has learned to capture the statistical structure of the text effectively, generalizing well to validation samples while avoiding memorization. The chosen balance of model depth, embedding size, and regularization made this configuration the most optimal among all tested setups

Underfit:

```
Epoch 1: Train Loss=4.2679 | Val Loss=3.7819 | Perplexity=43.90
Epoch 2: Train Loss=3.7373 | Val Loss=3.6310 | Perplexity=37.75
Epoch 3: Train Loss=3.5552 | Val Loss=3.5413 | Perplexity=34.51
Epoch 4: Train Loss=3.4738 | Val Loss=3.5220 | Perplexity=33.85
Epoch 5: Train Loss=3.4581 | Val Loss=3.5187 | Perplexity=33.74
Epoch 6: Train Loss=3.4552 | Val Loss=3.5178 | Perplexity=33.71
Epoch 7: Train Loss=3.4543 | Val Loss=3.5178 | Perplexity=33.71
Epoch 8: Train Loss=3.4539 | Val Loss=3.5171 | Perplexity=33.69
Epoch 9: Train Loss=3.4537 | Val Loss=3.5183 | Perplexity=33.73
Epoch 10: Train Loss=3.4536 | Val Loss=3.5163 | Perplexity=33.66
```



Model Configuration:

```
EMB_DIM = 64          # smaller embeddings
NHEAD = 2             # fewer attention heads
```

```
NLAYERS = 1          # only one layer
FF_DIM = 128         # tiny feedforward layer
DROPOUT = 0.5        # heavy regularization
```

In this configuration, the model exhibited clear signs of **underfitting** due to its limited representational capacity and strong regularization. The model used a small embedding size of 64, only two attention heads, a single Transformer layer, and a small feed-forward dimension of 128, combined with a high dropout rate of 0.5.

As shown in the loss curves, both training and validation losses decreased rapidly during the first few epochs but plateaued early (around 3.45 for training and 3.51 for validation). The early convergence and high final losses indicate that the model was too simple to effectively learn the underlying linguistic structure of the dataset.

8. Evaluation

Metric used: **Perplexity**

Perplexity= e^{Loss}

For the best-fit model:

- Validation loss decreased steadily
- Final perplexity was significantly lower than underfitting and overfitting runs
- Model generated consistent predictions during training

Training and validation curves were plotted for all three scenarios as required.