Re-implementation and Evaluation of the Transformer Model on a Small-Scale Dataset

Philip Pincencia
Department of Electrical and Computer Engineering
University of California, San Diego
A17333789

Raj Bapat
Department of Mathematics
University of California, San Diego
A18533089

April 16, 2025

Video Link: `https://www.youtube.com/watch?v=kFev-Gibx-k`

Abstract

The Transformer model has revolutionized natural language processing (NLP) by leveraging self-attention mechanisms that eliminate recurrence and enable efficient parallelization. In this project, we re-implement the Transformer architecture from scratch and evaluate its performance on a small-scale English-German translation dataset from the IWSLT 2017 corpus.

Our implementation incorporates all key components seen in *Attention Is All You Need* (Vaswani et al) such as masked attention, padding masks, teacher forcing during training, and autoregressive decoding during inference. We also record the metrics of our Transformer, measuring BLEU scores, token-level accuracy, and inference speed, and compare performance with the original Attention paper and LSTMs. Finally, we discuss the challenges of training Transformers on small datasets and propose potential solutions to improve generalization.

1 Introduction

Since its introduction, the Transformer model has become a cornerstone of NLP, outperforming recurrent neural networks (RNNs) and convolution-based models on various tasks, including machine translation and language modeling. Unlike RNN-based architectures (e.g., LSTMs), the Transformer utilizes self-attention

to capture long-range dependencies without recurrence, thereby enabling improved parallelization and potentially faster training.
However, Transformers typically excel on large-scale datasets. When trained on limited data, they tend to overfit and may struggle to outperform other models. The goal of this project is to investigate the behavior of Transformers under data-scarce conditions, compare them to the original Transformer paper as well as LSTM baselines, and explore ways to optimize their performance.

2 Problem Definition

This project addresses the following key challenges:

- **Transformer Re-implementation:** Implementing the Transformer model from scratch, including multi-head attention, masked self-attention, and encoder-decoder layers.
- **Small-Scale Training:** Evaluating the performance of Transformers on a limited dataset (a subset of IWSLT 2017), which is not their typical domain.
- **Mitigating Overfitting:** Applying techniques such as dropout, weight decay, and padding masks to improve generalization.
- **Efficient Decoding:** Using teacher forcing during training while adopting autoregressive decoding during inference.

3 Approach

3.1 Dataset and Preprocessing

We use the IWSLT 2017 English-German translation corpus , which was curated by IWSLT. The dataset comprises:

- **Training:** ∼200,000 sentence pairs
- **Validation:** ∼1,000 sentence pairs
- **Testing:** ∼1,000 sentence pairs

We tokenized our dataset using **Byte Pair Encoding (BPE)** with a maximum vocabulary size of 8,000, scaled down from 32,000 in the original paper. This reduction was necessary due to the smaller dataset size (**200,000 sentence pairs** compared to 4.5 million in the original implementation). A smaller vocabulary helps prevent overfitting and improves efficiency for training on limited data.

3.2 Model Architectures

3.2.1 Transformer Model

Architecture Diagram: Figure below shows the Transformer architecture diagram as described in the paper "Attention is All You Need".

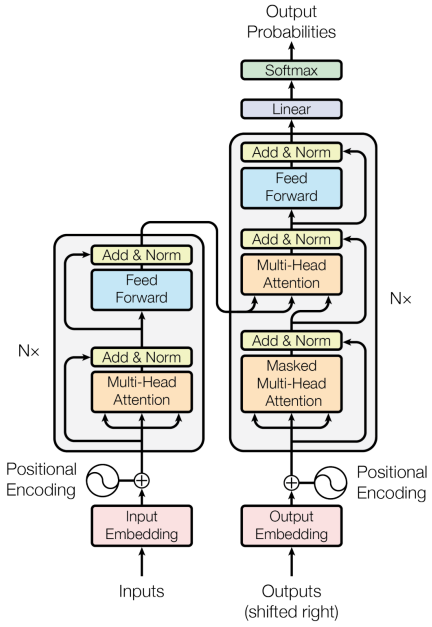

Figure 1: The Transformer - model architecture.

Our Transformer implementation follows the architecture of *Attention Is All You Need* (Vaswani et al). with some modifications:

- **Model Size:** Hidden size $d_{\text{model}} = 256$, chosen to reduce overfitting and meet our hardware constraints.
- **Attention Heads:** 4 heads to allow richer representations.
- **Depth:** 4 encoder and 4 decoder layers to keep the model lightweight.
- **Feed-Forward Dimension:** $d_{\text{ff}} = 4 \times 256 = 1024$.
- **Dropout:** 0.3 applied in attention and feed-forward sublayers.

Parameter Count: A rough calculation of the number of parameters is as follows:

- **Embeddings:** Two embedding layers with ≈4000 tokens each: $2 \times (4000 \times 256) \approx 2,048,000$ parameters

- **Encoder Layers:** Each layer has $\approx 789,760$ parameters; with 4 layers: $4 \times 789,760 \approx 3,159,040$ parameters
- **Decoder Layers:** Each layer has $\approx 1,053,440$ parameters; with 4 layers: $4 \times 1,053,440 \approx 4,213,760$ parameters
- **Final Projection:** A linear layer mapping from 256 to 4000 tokens: $256 \times 4000 \approx 1,024,000$ parameters

Thus, the total parameter count is roughly:

$$2,048,000 + 3,159,040 + 4,213,760 + 1,024,000 \approx 10.44 \text{ million parameters.}$$

3.2.2 Masked Attention and Padding Masks

Masked Attention: During training, we apply a mask to the decoder's self-attention to prevent it from "seeing" future words. This forces the model to rely only on the past and current tokens when predicting the next word. Without this masking, the decoder could trivially copy the target sequence, leading to unrealistic performance during training. It ensures that at timestep t, the model only attends to previous positions $\leq t$:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V,$$

where

- Q, K, V are Query, Key, and Value Matrices
- M is a mask matrix with $-\infty$ for positions $\leq t$ (i.e. upper triangular matrix)
- d_k is the dimension of the key vectors.

Padding Masks: Since sentences have different lengths, we pad shorter sequences to a fixed length. However, these padding tokens carry no meaning, so we use a padding mask to exclude them from attention calculations. This ensures that the model does not assign importance to padding tokens when computing attention scores.

3.2.3 LSTM Baseline

While our primary focus is on the Transformer, we also implemented a bidirectional LSTM model as a baseline. However, the remainder of this report focuses on the performance and re-implementation of our attention-based model since the LSTM model doesn't produce a meaningful BLEU result, although we will still include it for the sake of completeness

3.3 Training and Decoding Strategy

During training, the Transformer is fed the ground-truth target sequence shifted right by one token. This approach, known as **teacher forcing**, allows the model to converge faster and learn more stable alignments. Without teacher forcing, the model must rely on its own imperfect predictions early in training, leading to error accumulation.
Mathematically, if y_t is the ground truth token at time step t, and $\hat{y}_t$ is the predicted token, the decoder hidden state is updated as:

$$s_t = f(s_{t-1}, y_{t-1})$$

During inference, the model generates tokens one by one, feeding its own previous predictions back into the decoder. We use **autoregressive decoding**:

$$s_t = f(s_{t-1}, \hat{y}_{t-1})$$

This is called *autoregressive decoding* and is necessary since ground-truth tokens are not available at test time.
For both equations,

- $\boldsymbol{y}$ (*ground-truth token*): At each time step t, y_t is the token from the reference (gold) translation.
- $\boldsymbol{\hat{y}}$ (*predicted token*): $\hat{y}_t$ is the model's predicted token at time t. During inference, the model must rely on its own predictions rather than the ground truth.
- $\boldsymbol{s_t}$ (*decoder state*): This represents the internal hidden state (or context) of the decoder at time t. It contains information about all previously processed tokens (or states).
- $\boldsymbol{f}$ (*decoder update function*): A generic function that describes how the decoder updates its internal state. In the Transformer, f is implemented via multi-head attention and feed-forward layers. In other words, it takes the previous state s_{t-1} and an input token (either ground-truth y_{t-1} or predicted $\hat{y}_{t-1}$) to produce the new state s_t.

Why not use teacher forcing during inference? In real-world applications, we do not have access to ground-truth translations at test time. The model must generate outputs based on its previous predictions, making autoregressive decoding necessary for deployment.

4 Experiments and Results

4.1 Optimizer and Learning Rate Scheduler

We utilized the optimal/chosen parameters from the original Transformer paper, with weight decay to reduce overfitting:

- Learning rate: 0.0007
- Betas: (0.9, 0.98)
- Epsilon: 1×10^{-6}
- Weight decay: 0.001

We also use a warm-up schedule followed by linear decay, as described in the paper, to stabilize training. These hyperparameters were directly adapted from the original work to ensure reproducibility.

4.2 Hardware and Performance

All experiments were conducted with the following specifications:

- **Machine type:** `g2-standard-4` (4 vCPUs, 16 GB Memory)
- **GPU:** 1 x NVIDIA L4 (with 24 GB VRAM)
- **CPU platform:** x86/64

4.3 Evaluation Metrics

Performance is measured using:

- **BLEU Score:** Measures n-gram overlap between generated and reference translations.

$$\text{BLEU} = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \times \min(1, e^{1-\frac{r}{c}})$$

- **Token-Level Accuracy:** Percentage of correctly predicted tokens (excluding padding). Token accuracy is computed as:

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} \mathbf{1}_{(\hat{y}_i = y_i)}}{N}$$

  where N is the total number of tokens in the dataset and $\mathbf{1}_{(\hat{y}_i = y_i)}$ is an indicator function that counts correct predictions.

- **Inference Speed:** Tokens processed per second during testing. To measure the model's inference speed, we:

  1. Ran the trained model on our test set in **autoregressive mode**, generating translations token-by-token for each sentence.
  2. Recorded the total number of tokens produced during inference (summing across all test sentences).

3. Measured the wall-clock time from the start of decoding until all sentences were completed.
4. Computed **tokens per second (tok/s)** as:

$$\text{Inference Speed} = \frac{\text{Total Decoded Tokens}}{\text{Total Inference Time (seconds)}}.$$

We obtained an approximate speed of **2200 tok/s**, reflecting the advantage of parallelized attention compared to strictly sequential models.

4.4 Results

Table 1: Performance Comparison of Transformer

Model	Test BLEU	Token Accuracy (%)	Inference Speed (tok/s)
Transformer	24	30.5	2200

5 Model Training Metrics and Performance Analysis

To evaluate the performance of our model, we tracked loss, accuracy, and BLEU score over epochs. These metrics help us understand how well our model is learning and generalizing to unseen data.

5.1 Loss over Epochs

Figure 2 shows the training and validation loss across epochs. We observe that:

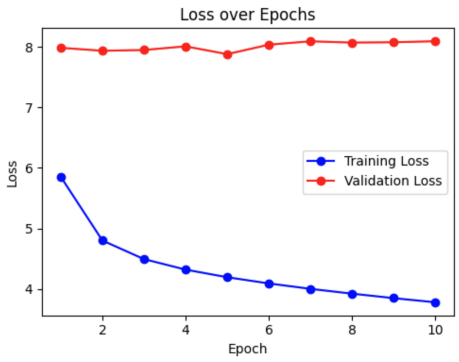

Figure 2: Loss over Epochs

The **training loss** (blue) steadily decreases, indicating that the model is learning from the data. The **validation loss** (red), however, remains high and does not show significant improvement. This suggests that while the model is optimizing on the training set, it struggles to generalize well to unseen data, potentially due to overfitting.
We trained our model using the **cross-entropy loss** with a label smoothing value of 0.1, following the original paper. Label smoothing helps prevent the model from becoming overconfident in its predictions by redistributing a small portion of the probability mass from the correct label to the incorrect ones. This technique improves generalization and stabilizes training.

5.2 Accuracy over Epochs

Figure 3 shows the accuracy for both training and validation sets. Key observations:

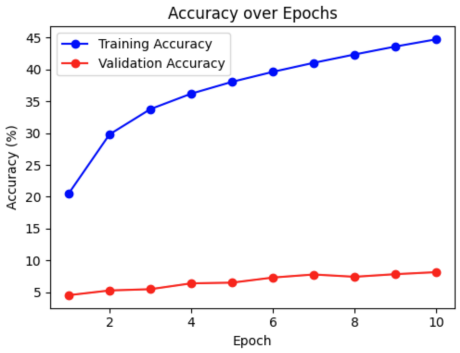

Figure 3: Accuracy over Epochs

Training accuracy improves consistently, reaching around **42.36%** at epoch 8. **Validation accuracy**, however, remains low (˜7.43%). This reinforces the earlier hypothesis that the model is overfitting to the training set.

5.3 BLEU Score Over Epochs

Figure 4 tracks the **BLEU score**, a metric used to evaluate the quality of generated sequences in comparison to reference texts. Notable trends: BLEU score fluctuates across epochs, with a peak at **epoch 8 (24.12)** before declining. This suggests that epoch 8 provides the best trade-off between training progress and generalization.

5.4 Best Model Selection

Based on the observations above, we selected **epoch 8** as the best model checkpoint since it achieves the highest BLEU score (24.12) while maintaining

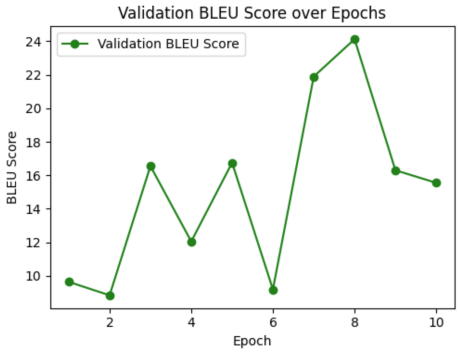

Figure 4: BLEU Score over Epochs

reasonable training accuracy. The summary of metrics at this epoch is provided in Table 2.

Train Loss	Train Accuracy (%)	Validation Loss	Validation Accuracy (%)	Validation BLEU
3.9204	42.36	8.0733	7.43	24.12

Table 2: Best model performance at epoch 8.

6 Conclusion and Future Work

Our final model achieved a BLEU score of 24 on the 200,000-sentence subset of the IWSLT 2017 English-German dataset, compared to the original paper's 26.4 BLEU trained on 4.5 million sentence pairs. While our model is significantly smaller, it performs competitively, demonstrating that efficient scaling can maintain strong translation quality even on limited data.

However, a key limitation is generalization. While our model performs well on the IWSLT dataset, it may struggle with more diverse corpora due to its reduced number of layers, heads, and model dimension. The original paper's model (8 heads, 6 encoder/decoder layers, 512-dimension) was trained on a much larger dataset and likely generalizes better to unseen translations.

6.1 Future Work

To further improve performance, we could scale up our model closer to the original paper's architecture, train on a larger dataset to enhance generalization, and explore techniques such as low-rank factorization or quantization to balance model size with performance.

Expanding our dataset and computational resources would allow us to bridge the gap between our model and state-of-the-art translation systems.

References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). "Attention Is All You Need." Advances in Neural Information Processing Systems (NeurIPS).

Cettolo, M., Federico, M., Bentivogli, L., Niehues, J., Stüker, S., Sudoh, K., Yoshino, K., & Federmann, C. (2017). "Overview of the IWSLT 2017 Evaluation Campaign." In Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT 2017), Tokyo, Japan, December 14–15, 2017, pp. 2–14.