Heinrich Heine
University
Düsseldorf

Training Report

Leon Markwart

Submission: 29th January 2025

Supervisor: Dr. Carel van Niekerk and Hsien-Chin Lin

Contents

List of Figures

List of Tables

Listings

Chapter 1.

Do it yourself: Transformer

The Transformer is a landmark model introduced by [Vas+17] in their paper **“Attention is All You Need” (2017)**. This model initially revolutionized natural language processing (NLP) by discarding traditional recurrent architectures like RNNs and LSTMs, but also found its way into other modalities like computer vision, where the variation "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" by [Dos+20] also outruled the CNN-architectures in several tasks.

The strength of the transformer specially lies in the **Attention** mechanism, which is the core innovation that allows the model to dynamically focus on the most relevant parts of the input sequence, regardless of its position. The Attention mechanism helps capture long-range dependencies and relationships between tokens, which traditional models like RNNs and CNNs struggled to manage effectively. Furthermore, it allows for high parallelization and thus efficent training, wich is crucial for models of that size.

1.0.1. Positional Encoding

The paper introduces positional encodings to inject information about the position of tokens in a sequence into the Transformer model. These encodings are added to the input embeddings before being passed to the model.

The paper defines positional encodings as:

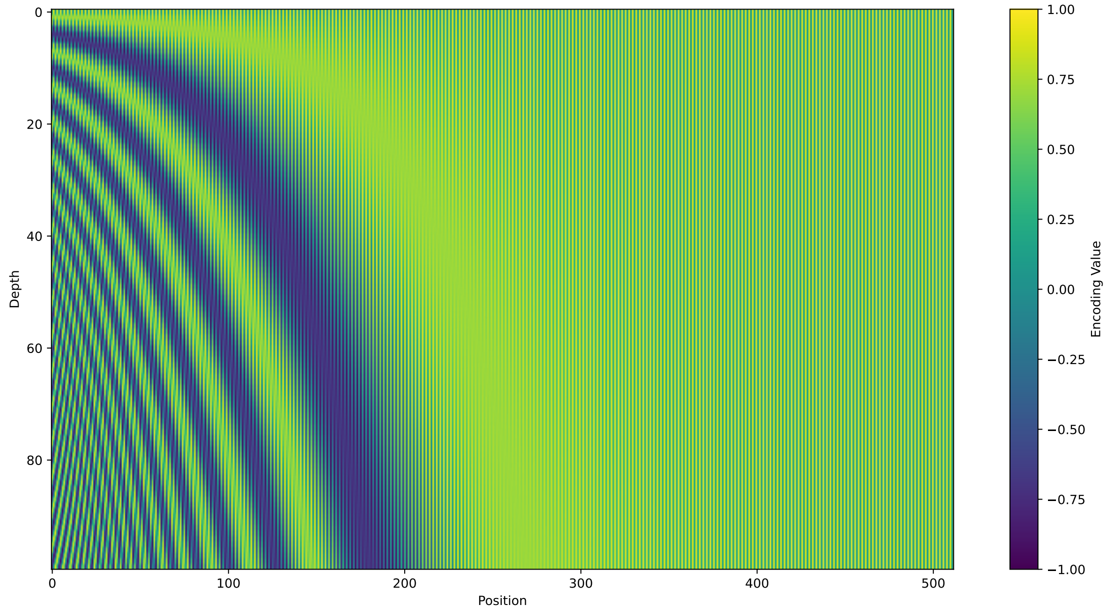

Figure 1.1.: This is a visualization the positional encoding values. This figure showcases the first 100 positions for a transformer model of depth 512. The image is self generated.

$$PE_{t,2i} = \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right), \quad PE_{t,2i+1} = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right),$$

where t is the position, i is the dimension index, and d is the dimensionality of the model.

Counterintuitively, the transformer network is **permutation invariant**, meaning that the token representations can be permuted after adding the positional encoding and the model is still able to determine it's original order. The proof, that any two positional ancodings are a linear function of each other and that the wavelengths form a geometric progression can be found in the appendix A.

1.0.2. BPE

Byte Pair Encoding (BPE) is a subword tokenization technique that splits text into smaller units, such as subwords or characters, to handle rare or unseen words effectively. It was initially introduced in data compression but also found it's way into natural language processing.

Exception in encoding numbers

While BPE encoded tokens are well accesible for LLM's, the encoding of digits negatively impacts a models arithmetic understanding. A recent study by [SS24] suggests that encoding a number from right to left leads to large improvements of a models arithmetic understanding.

1.0.3. Embedding

Embedding is a learnable map from the tokens into a dense, continuous-valued vectors in a fixed-dimensional space. These embeddings capture semantic and syntactic relationships between tokens.

In the transformer model embeddings are used in the encoder and decoder after the positional encoding. The inverse map of the embedding is used to map back the representations into the tokens. It is highly beneficial if the embedding and the inverse embedding have shared parameters. Also, for a shared tokeization encoder it can also be beneficial if encoder and decoder embeddings are shared.

1.0.4. Attention

The attention mechanism is at the heart of the Transformer architecture. It allows the model to selectively focus on the most relevant parts of an input sequence, helping it understand relationships between elements in a context-aware way. This ability, while computationaly expensive, makes the Transformer very flexible (low inductive bias).

Self-Attention

Self-attention enables the model to look at all parts of an input sequence and determine which parts are most relevant to each other. For example, when processing a sentence, the mechanism figures out how words relate to one another, regardless of their position. This creates richer, context-aware representations of each word. In essence, self-attention gives the model a "big picture" view, where every element can interact with every other.

Multi-Head Self-Attention

Multi-head self-attention enriches self-attention by using multiple "heads," each analysing on different aspects of the input. Think of it like having multiple perspectives on the same data: one head might capture short-range dependencies , while another focuses on long-range connections. The results from these heads are concatenated, giving the model a more nuanced understanding of the sequence.

Cross-Attention

Cross-attention is used in the Transformer decoder to align two sequences (can also be **different modalities**, such as an input sentence (source) and a translated sentence (target). While self-attention focuses on relationships within a single sequence, cross-attention aligns the target sequence with the most relevant parts of the source. This ensures that the output sequence is generated with proper context from the input, enabling accurate translations or predictions.

1.0.5. Layer Normalization

Layer Normalization maps a distribution into a certain range. It's parameters γ and β are excluded from weight decay because they control the normalization process, not the model's capacity to fit the data.

$$\mathrm{LN}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta,$$

Chapter 2.

Training

In the following section, the training pipeline, final training results and (statistically unsignificant) ablation studies, that led to the final training configuration will be presented.

Throughout the whole training, the wmt17[1]-dataset has been used. The dataset consists of 5.91 million training samples for the german-english translations. After filtering, the size rapidly shrunk to roughly **0.7 million samples**.

2.0.1. Training setup

Tokenizer

The tokenizer has been trained on **both languages** in the dataset, wich makes it vocabulary shared. The **vocab size** has been set to 30.016 in the following experiments. The length of all tokenized translations in the training dataset (figure 2.1) suggests that most tokenization lengths are below 30 tokens. The **maximal token length** of the transformer model has therefore been set to 64 tokens.

Remarkable about the german and english tokenization in figure 2.1 is that while the untokenized translation in german is on average significantly longer than in english, their embeddings are roughly equal distributed.

[1] `https://huggingface.co/datasets/wmt/wmt17`

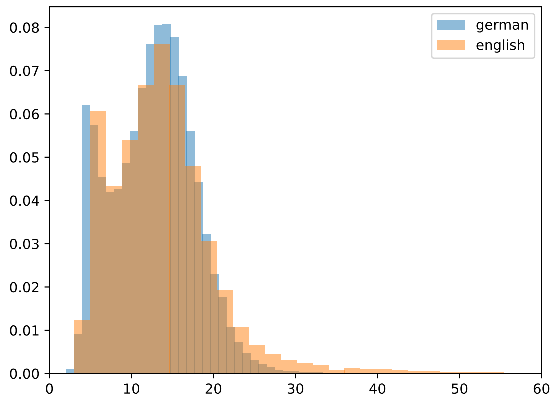

Figure 2.1.: The token size of the vocabulary barely exceeds 30 for the train split of the dataset

Data Filtering and Cleaning

The training pipeline has been easy and computationally efficient implemented by the Huggingface Dataset Preprocessing[2] functions. Their caching abilities allow the computationally heavy operations to be run only once and be reloaded on ervery additional filtering. In my implementation **use_costum_dataset()**, the flags *split* (train/validation/test), *tokenize* (if the dataset should be tokenized or not), and *src_bos* (if the source should start with [BOS]) are available.

Maximum Learning Rate Calculation

The learning rate scheduler from "Attention is all you need" came with a predefined learning rate curve. Although the scheduler is proportional to the initial learning rate, it has no option to set the maximum learning rate in the curve. I extended the scheduler to be multilied by a scale, to optinally set a maximum learning rate where the scheduler peaks. Different learning rates will be compared in section 2.0.2.

$$\text{scale} = \frac{\text{max\_lr} \cdot \sqrt{d_{\text{model}} \cdot \text{warmup\_steps}}}{\text{base\_lr}}$$

[2] `https://huggingface.co/docs/datasets/use_dataset`

Pytorch Lightning

Pytorch Lightning[3] is a framework to abstract repetetive boilerplate code. It consists of two main parts: The **LightningModule**, where a model and the training loop is defined, and the **Trainer** class, where different strategies and modular interchangable options can be tried. Some of the tasks that can be reduced are:

- Defining train/val/test loop, including optimizer- and scheduler-call
- Reporting loss, metric and test samples to loggers like Tensorboard
- Checkpointing with conditions
- Device mapping
- Hyperparameter documentation
- Training with fraction of data
- Training strategies like gradient accumulation, scaling, clipping, mixed precision, ...
- and much more

The explained pseudocode of the LightningModule can be found in the appendix B.

2.0.2. Ablation Studies

In the following section some experiment settings are compared. The comparisons are by no means statistically significant, since this would require multiple runs, to overcome observations caused by different ransom initializations. For every of the following comparisons, only one hyperparameter is changed, all other remain the same, what ensures a fair comparison. The following hyperparameters has been used:

[3] `https://lightning.ai/docs/pytorch/stable/starter/introduction.html`

Hyperparameter	**Value**	**Hyperparameter**	**Value**
batch size	64	max_len	64
d_{model}	512	max_lr	Ablation 2.0.2
$dim_{feedforward}$	2048	n_{heads}	8
dropout	0.1	num_decoder_layers	6
label smoothing	0.1	num_encoder_layers	6
BOS in source	Ablation 2.0.2	vocab_size	30016
warmup_steps	4000	fraction_train	0.25

Table 2.1.: Hyperparameters for the model.

Ablation 1: Leading [BOS] in Source Input

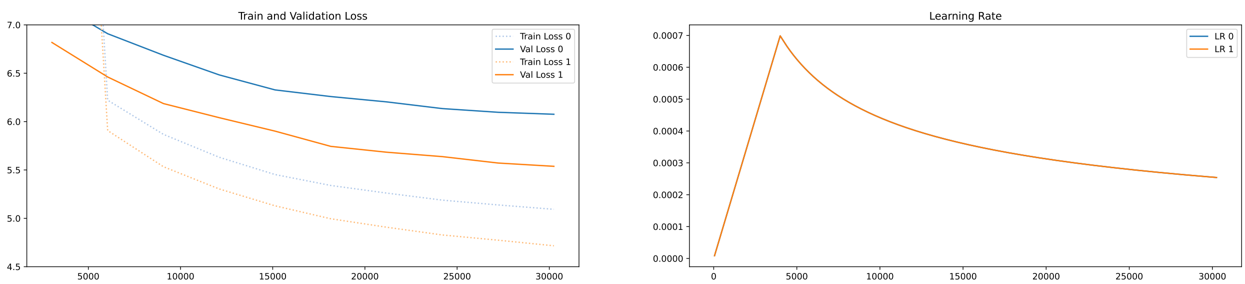

Figure 2.2.: Run 0, including the [BOS]-token is performing worse than Run 0, excluding the token.

The first pairwise comparison is wether a leading [BOS]-token (Beginn of sentence) is benefitial for training. While a leading [BOS]-token in the target input is required for the training to not provide the model informations about the token it should predict, this doesn't seem to obvious for the source input. For the training curve in figure 2.2, **Run 0** includes the [BOS]-token in the source sentence, while **Run 1** does not. The version without the initial token does significantly better that the version that includes it.

Ablation 2: Different Learning Rates

Like described in 2.0.1, the scheduler is extended by the *max_lr* option. This is usefull because [Smi15] proposed a methode to get the maximal liable learning rate in a short run, wich is also implemented into pytorch lightning. In figure 2.3 the learning rate finder suggests a point that is close to the proposed *max_lr* in the paper "Attention is all you need", approximately $7 \cdot 10^{-4}$.

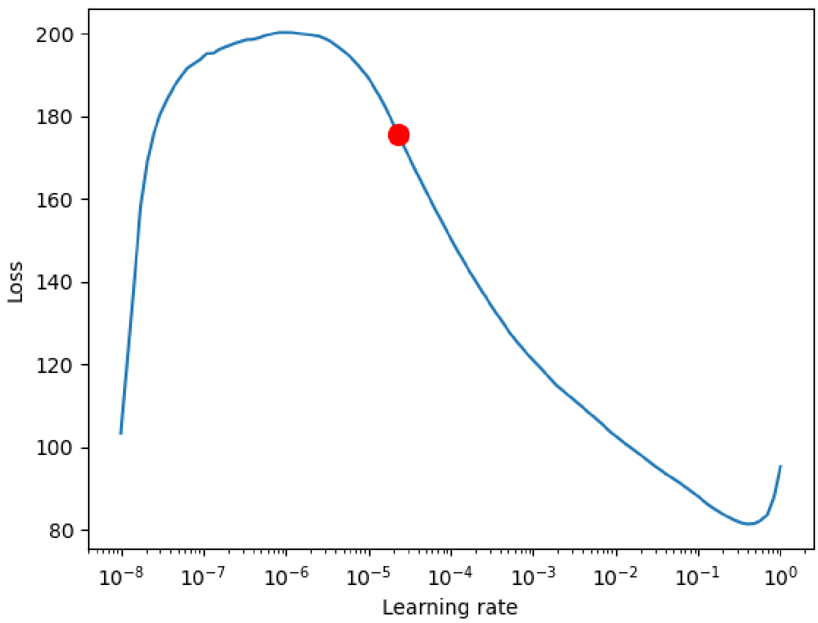

Figure 2.3.: The learning rate is started at 10^{-8} and is increased with every step. The red dot marks the steepest gradient, wich is the suggested learning rate. For learning rates above 10^{-1} the loss starts to diverge. That learning rates below 10^{-6} are improving is a bug.

In the next experiment, three different learning rate schedules are compared to each other. The first learning rate curve is lower than the second (the original learning rate scheduler) curve and the third curve is a litte higher. Out of all three in figure 2.4 compared learning rates, the one used in "Attention is all you need" outperforms both, the slightly higher and the lower learning rate.

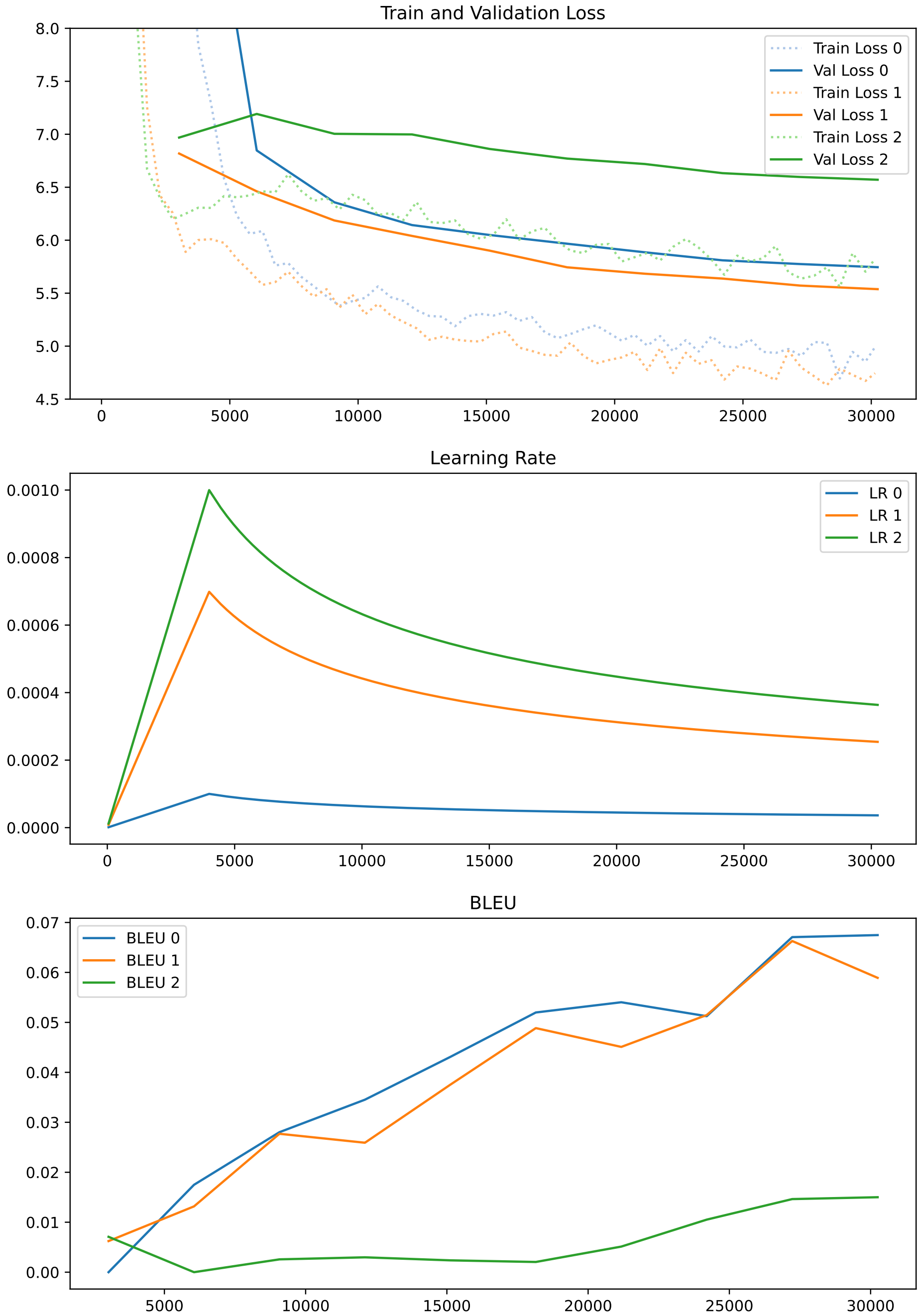

Figure 2.4.: Maximal LR: Curve 0: 10^{-4} Curve 1: $7 \cdot 10^{-4}$ Curve 2: 10^{-3}

2.0.3. Final Training

With a suitable set of hyperparameters identified in smaller experiments, it is now time to scale up the model. Due to computational constraints, only the number of attention heads in the encoder and decoder has been increased to 8 heads (6 heads in the ablation studies).

BOS in source = **True** $\qquad\qquad max_{lr} \approx \mathbf{7 \cdot 10^{-4}}$

For hyperparameter tuning, a reduced model and dataset fraction were used. In the main run, training will be scaled up with the following adjustments:

num_encoder_layer = **8** $\qquad$ num_decoder_layer = **8** $\qquad$ fraction_train = **1.0**

The following figure is the train/validation loss, the learning rate scheduler and the bleu score of the final training run. Since mixed precision is not yet implemented in pytorch lightning (while it is in pytorch), all floating numbers are of type **float32**. One batch consists of **747.464 samples**, wich gives 12.101 batches with a batchsize of 16. One batch, run on a M1 MacBook Pro with 16GB of shared memory was running for **3:12 hours per epoch**. The training was ended after 14 epochs, wich made a total training time of **44 hours**, due to time limitations. The validation loss in figure 2.5 has not yet fully converged and the checkpoint allows to resume training, potentially still increasing the bleu validation score.

The test set's bleu score of the best (last) checkpoint is 0.1072. The first samples of the test-set and their greedy predictions are to be found in the appendix A.

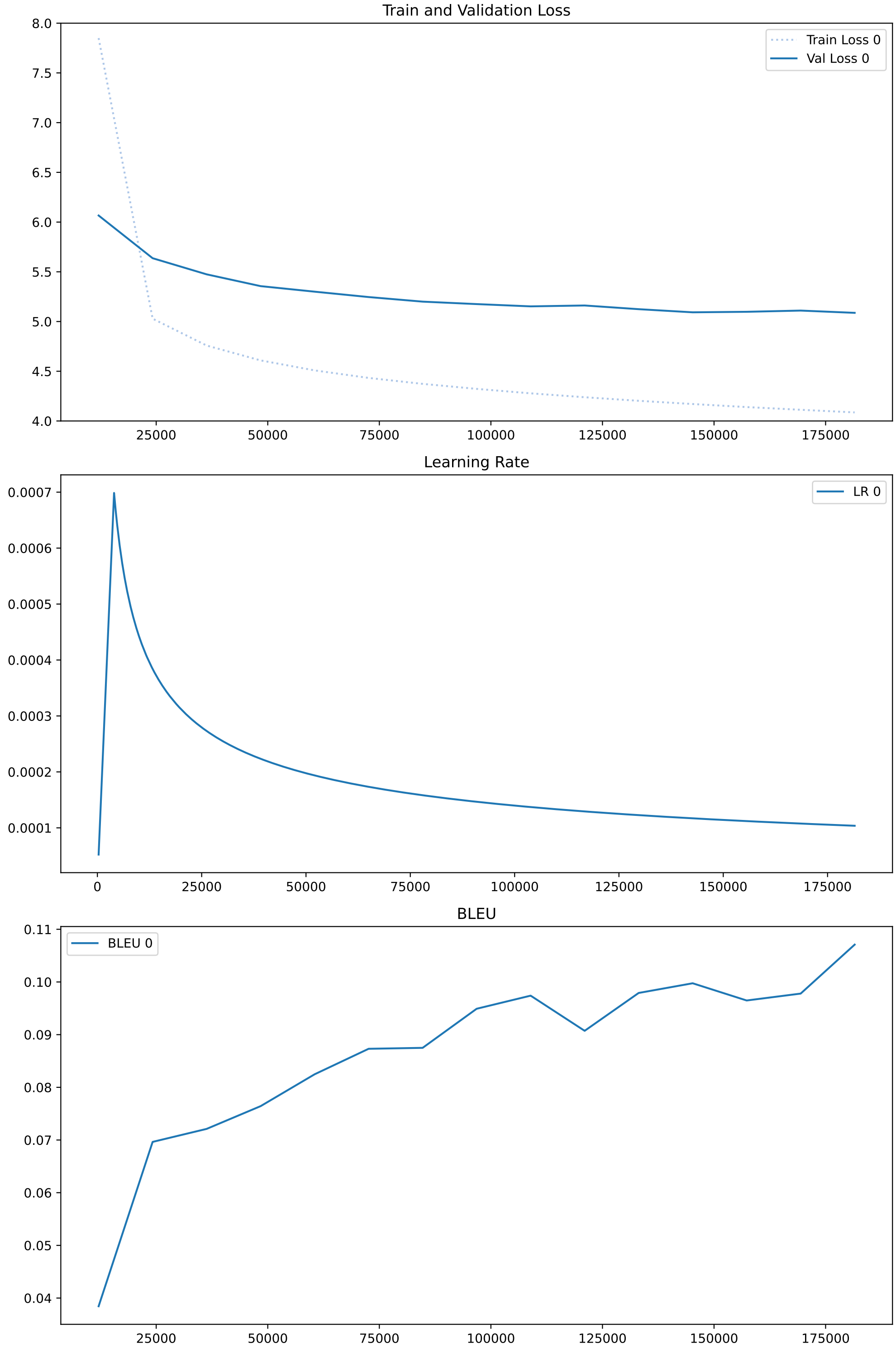

Figure 2.5.: The final training curve after almost 2 days of training. The model is not fully converged, but time was running out. The bleu score is calculated for the validation set at every epoch.

Chapter 3.

Summary

The report provides a comprehensive overview of the Transformer model, originally introduced in *Attention is All You Need* (2017), highlighting its core innovation—the attention mechanism. Unlike traditional RNNs and CNNs, Transformers leverage self-attention to efficiently capture long-range dependencies and enable parallel computation. A finding for the BPE was, that **while german demands more chars** than the english translation, both **encodings are on average of the same length**.

The hyperparameter search highlighted both effective and ineffective training strategies. The first major finding was that, contrary to some blog posts, the **initial [BOS] token before the source input is not necessary** and, in this case, even negatively impacted the training curve. The second key finding was that the **learning rate scheduler from the original paper is highly optimized**, outperforming both slightly higher and lower learning rates.

For the final training, the model architecture was scaled up with eight encoder and decoder layers, incorporating the results of the hyperparameter search. While the full-scale training performed better than the downscaled hyperparameter study, it **still did not achieve a BLEU score high enough for a practical translation model**. Limited computational power and the reduced amount of training data after filtering prevented the training of a larger model.

Word Count: **2053**

Appendix A.

Equations

1.: Two positional encodings are a linear function of each other

Definition of PE :

$$PE_{t+k,2i} = \sin\left(\frac{t+k}{10000^{\frac{2i}{d}}}\right),$$
$$PE_{t+k,2i+1} = \cos\left(\frac{t+k}{10000^{\frac{2i}{d}}}\right).$$

Express $PE_{t+k,2i}$:

Using the sum of angles formula:

$$\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b),$$

$$PE_{t+k,2i} = \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right)\cos\left(\frac{k}{10000^{\frac{2i}{d}}}\right) + \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right)\sin\left(\frac{k}{10000^{\frac{2i}{d}}}\right).$$

Express $PE_{t+k,2i+1}$:

Using the cosine sum formula:

$$\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b),$$

$$PE_{t+k,2i+1} = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right)\cos\left(\frac{k}{10000^{\frac{2i}{d}}}\right) - \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right)\sin\left(\frac{k}{10000^{\frac{2i}{d}}}\right)$$

Linear Representation:

Let:

$$a = \cos\left(\frac{k}{10000^{\frac{2i}{d}}}\right), \quad b = \sin\left(\frac{k}{10000^{\frac{2i}{d}}}\right).$$

Substitute a and b into the equations:

$$PE_{t+k,2i} = a \cdot PE_{t,2i} + b \cdot PE_{t,2i+1},$$

$$PE_{t+k,2i+1} = a \cdot PE_{t,2i+1} - b \cdot PE_{t,2i}.$$

Thus, PE_{t+k} can be expressed as a linear transformation of PE_t using the coefficients a and b , which depend only on k.

2.: The wavelenghts form a geometric progression from 2π to $10000 \cdot 2\pi$

Wavelength Definition: The frequency of each positional encoding dimension is inversely proportional to $10000^{\frac{2i}{d}}$:

$$\text{Frequency} = \frac{1}{10000^{\frac{2i}{d}}}.$$

The corresponding wavelength λ is the reciprocal of the frequency:

$$\lambda = 10000^{\frac{2i}{d}}.$$

Wavelength Progression: For $i = 0$, the smallest wavelength is:

$$\lambda_{\text{min}} = 10000^{\frac{0}{d}} = 1.$$

For $i = d/2 - 1$ (the largest i in d -dimensional space):

$$\lambda_{\text{max}} = 10000^{\frac{2(d/2-1)}{d}} = 10000^{1-\frac{2}{d}}.$$

The ratio of successive wavelengths is constant:

$$\text{Ratio} = \frac{\lambda_{i+1}}{\lambda_i} = \frac{10000^{\frac{2(i+1)}{d}}}{10000^{\frac{2i}{d}}} = 10000^{\frac{2}{d}}.$$

This confirms that the wavelengths form a geometric progression with a common ratio of $10000^{\frac{2}{d}}$, ranging from 2 to $10000 \cdot 2$.

Appendix B.

Listings

B.1. Pytorch LightningModule

Listing B.1: All nessasary functions of LightningModule

```python
import pytorch_lightning as pl

class TransformerModel(pl.LightningModule):

    def __init__(self, ... ):
        """
        Here, the model, loss and matrics can be defined.
        Saves all the args/kwargs in hparams.yaml for reproducability.
        """
        self.save_hyperparameters()

    def forward(self, src_input, tgt_input):
        # This is the forward call, whenever the model is directly called

    def training_step(self, batch, batch_idx):
        """This is the training loop. It should return a step's loss and report the loss."""
        prediction = self(...)  # calls forward()
        self.log('train_loss', loss_step, prog_bar=True, on_epoch=True, on_step=True, logger=True)
        # This reports the training status to or are multiple loggers

        return loss_step

    def validation_step(self, batch, batch_idx):
        """This is the validation loop. Reporting loss, metrics and samples should be done here"""
        return loss_step

    def test_step(self, batch, batch_idx):
```

```
        """Equivalent to the validation step, but never gets called while training"""


    def configure_optimizers(self):
        """Optimizers and schedulers can be defined here"""
        return {
            'optimizer': optimizer,
            'lr_scheduler': {
                'scheduler': scheduler,
                'interval': 'step',
                'frequency': 1,
            }
        }

    def {train|val|test}_dataloader(self):
        """
        Returns train/val/test_loader
        Can also be added modular in the trainer.fit(model, ...loader) method
        """
        ...
```

Appendix A.

Tables

Table A.1.: The first 23 test samples and greedy predictions of the filtered translation dataset.

src_input	tgt_output	prediction
28-jähriger Koch in San Francisco Mall tot aufgefunden	28-Year-Old Chef Found Dead at San Francisco Mall	28-yearthless-search-bine in San Francisco
Er war ein freundlicher Mensch mit einem großen Herzen.	He was a kind spirit with a big heart.	He was a great success.
Er war der Bruder, der mit dem Strom schwamm.	He was the brother that went with the flow.	He was the brother to be with the electricity.
Jennifer Aniston muss nicht immer perfekt oder erfolgreich sein.	Jennifer Aniston need not always be perfect or successful.	Jennifer Aniston must not always be perfect or successful.
Der Film läuft bei uns ab dem 25. August.	The film is released in Germany on 25 August.	The film runs at our own level of the 25th of
Golfer Langer erhält die Sportpyramide	Golfer Langer is awarded the Sport Pyramid	Grosser Langer receives the sportspyramide
Seine Erfahrungen auf dem Pferd sind überschaubar.	His experience on horseback is negligible.	Ladies and gentlemen,
Für den 58-Jährigen war es eine Premiere.	It was a first for the 58-year-old.	Further information is available on the 58-year floor.
CHIO: "Goldene Sportpyramide" für Bernhard Langer	CHIO: "Golden Sport Pyramid" for Bernhard Langer	CHIO: "G volle Sportpyramide" for Bernhard Langer
Ein Kilometer bei diesem Tempo, ich habe Angst gehabt.	After a kilometre at this speed I was scared.	A km at this pace, I have fear.
Und damit endete die Reitkarriere dann auch schon wieder.	And with that, his riding career came to an end again.	And that is where the Commission is going to be a little again.
Der Grund war durchaus überzeugend.	He had been thoroughly convinced.	The reason was quite clear.
An Auszeichnungen mangelt es dem sympathischen Sportler nicht.	The friendly sportsman is not lacking for awards.	Annexes is not lacking to the sympathical sport.
Selbst die britische Queen hat ihn schon geadelt.	Even the British Queen has bestowed an honour upon him.	Of course, the British meters him.
Langer ist der 18. Preisträger der Sportpyramide.	Langer is the 18th person to be awarded the Sport Pyramid.	Langer is the 18th winners of sport sport.
Seit Jahrzehnten fördert Langer den Nachwuchs.	Langer has been encouraging up-and-coming talent for years.	Our company has been a member of the Langer.
Bernhard Langer hielt Abstand zu den großen Tieren.	Bernhard Langer kept his distance from the large animals.	Bernhard Langered a new role to the great animals
Der, über die Rolle des Staates.	This is the role of the state.	They are not in the role of the
Er selber dagegen wäre ganz anders, behauptet Trump.	He would do things totally differently, Trump says.	He ourselves voted against.
Deshalb würden die Steuern für die Reichen erhöht, sagt sie.	This would increase taxes for the rich, she says.	That is why taxes would be needed for the review of the repr
Auf Trump könnten also noch parteiinterne Diskussionen zukommen.	Consequently, Trump could lead to discussions within the party.	On Trump, therefore, there could be a long deal of discussion.
Ihn will sie auf 15 Dollar pro Stunde erhöhen.	She wants to raise it to 15 dollars an hour.	Your room is not going to go into 15 Dollar per hour
Wenn sie in Deutschland ankommen, sind sie oft traumatisiert.	When they arrive in Germany, they are often traumatised.	When they are in Germany, they are in a very high way.

Bibliography

[Dos+20] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[Smi15] Leslie N. Smith. "No More Pesky Learning Rate Guessing Games". In: *CoRR* abs/1506.01186 (2015). arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186.

[SS24] Aaditya K. Singh and DJ Strouse. *Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs*. 2024. arXiv: 2402.14903 [cs.CL]. URL: https://arxiv.org/abs/2402.14903.

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Declaration on the use of AI

While creating the proposed work, following tools have helped in the following domains:

- **DeepL**: For special english translations
- **Github Copilot**: For programming, especially diagrams and latex
- **Perplexity**: Finding scientific references