Report for the Software Project
Build Your Own Transformer

Laurent Hug
Saarland University
Saarbrücken, Germany
s8lahugg@stud.uni-saarland.de

Abstract

The goal of this software project was to implement our own PyTorch version of the transformer architecture described in the iconic paper "Attention is All You Need" (Vaswani et al., 2017). This report describes the data set used to train the transformer, as well as the experiments done to improve performance. The project was quite successful, achieving a final Bleu score of 39.4 on the Multi30k data set. This result is competitive with state of the art machine translation models that were trained on this data at the First Conference on Machine Translation in 2016 (Specia et al., 2016).

1 Introduction

The paper "Attention is All You Need" (Vaswani et al., 2017) has single-handedly transformed the machine translation landscape since its introduction in 2017. By doing away with the old recurrent architecture, and completely relying on the attention mechanism, it was able to create a machine learning model that produces accurate sequence-to-sequence results, while being quick to train. The goal of this software project was to recreate this transformer architecture and apply it to a machine translation task. The task chosen by me was German-to-English translation, as this made it easy for me to understand and check the accuracy of the results.

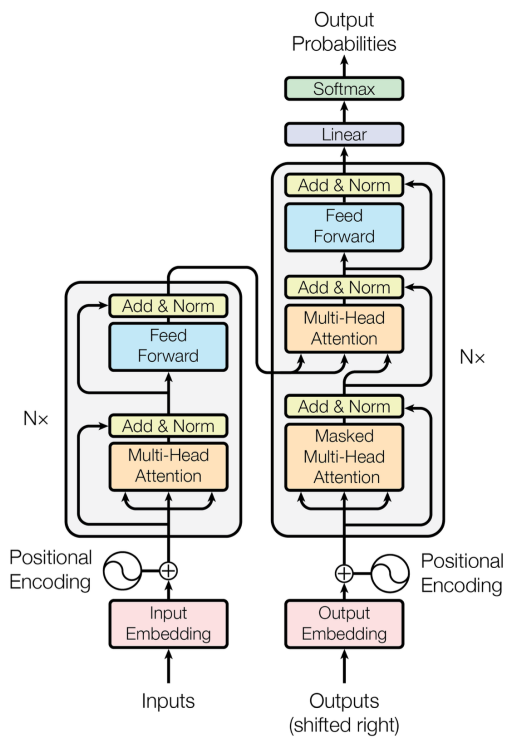


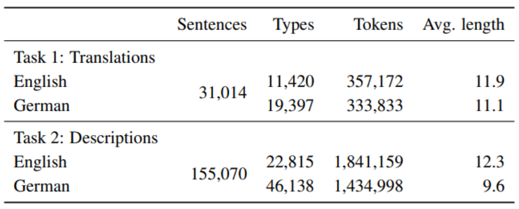
Figure 1: The Transformer - model architecture.

Figure 1: The Architecture of the transformer. Image taken from (Vaswani et al., 2017)

The implementation was done in Python using only basic PyTorch functions. I attempted to stay as close as possible to the architecture described in the original paper. As an advanced feature, I chose to implement the relative positional embedding as well as the ability to use multiple layers of encoder and decoder blocks.

2 Data

The data set I used to train and evaluate my transformer model was the Multi30k data set from the torchtext data set library. This data set is an extension of the well known Flickr30K data set, and was first introduced during the ACL 2016 machine translation conference in Berlin. One fifth of the Flickr30k data set was randomly selected and translated by a professional translator, in order to create a large corpus of sentence pairs while maintaining maximum sentence similarity. The corpus is made up of 31014 English-German sentence pairs of Flickr image descriptions. A comprehensive view of the data sets features can be found in Figure 2.



	Sentences	Types	Tokens	Avg. length
Task 1: Translations				
English	31,014	11,420	357,172	11.9
German		19,397	333,833	11.1
Task 2: Descriptions				
English	155,070	22,815	1,841,159	12.3
German		46,138	1,434,998	9.6

Figure 2: Description of the Multi30k data set used to train and evaluate the transformer. Image taken from (Specia et al., 2016)

The data was split into training set, validation set and test set according to the default split provided by torchtext. This resulted in 29000 training sentence pairs, 1014 validation sentence pairs and 1000 test sentence pairs.

As a preprocessing step the sentences were all changed to lower case, but punctuation was kept in tact. The vocabulary was built using only the training data, keeping both validation and test data untouched. Tokenization was and vocabulary building was done using spaCy and torchtext. One of the weak points of the data set is that it only contains sentences that are image descriptions. This results in major flaws in the final translation model that are further discussed in section 4.

3 Experiments

In order to train and test the transformer architecture, multiple experiments were run. First, the model was trained using the best hyperparameters found in the ”Attentionn is All You Need” paper (Vaswani et al., 2017) in order to get a baseline result. Second, the hyperparameters were further tuned using different techniques. A grid search was implemented over 48 different hyperparameter combinations and an attempt was made to further improve results by hand. Lastly, a model was trained over a larger number of epochs using the best found hyperparameter combination in order to produce the final results.

3.1 Set Up

The experiments were all done using my personal computer on a NVIDIA RTX 3070 GPU with 8GB of memory. Using this simple setup, each epoch over the training set took approximately 1 to 2 minutes. Before each experiment, it was checked that no sentences contained in the training set were also present in either the validation set or the test set. This was done so that no data could bleed into the training set and therefore affect performance. Any duplicate sentences were also removed in order to minimize any possible bias. The vocabulary also excluded any words that only appeared once in order to reduce its size on rare words. All experiments were done using Adam optimizer (Kingma and Ba, 2017) and cross entropy loss.

The biggest challenge in creating a good machine translation model was tuning the hyperparameters. The transformer architecture, like all machine learning architectures, has a number of these that can affect the model performance in a large way. In my case, I chose the following hyperparameters to tune the model. The sentence embedding size d_{model}, the number of multi-head-attention heads n_{heads}, the initial learning rate α and the amount of encoder/decoder layers n_{layer}. The feed forward expansion factor was always set to 4, do that the feed-forward hidden size d_{ff} was always four times as large as d_{model}. The model also implemented dropout after each sub-layer, before the result is added to the input. The dropout rate was also fixed at $d = 0.1$ and not further tuned. Lastly, the batch size was also fixed for all experiments at 32 and not tuned any more.

3.2 Baseline Model

As a first starting point, a machine translation model was trained using the best hyperparameters suggested in ”Attention is ALL You Need”. These were $d_{\text{model}} = 1024$, $n_{\text{heads}} = 16$, $\alpha = 0.0003$, $d_{\text{ff}} = 4096$ and $n_{\text{layer}} = 6$. Using these values, the model was trained for 15 epochs. After every training epoch, the validation set was used to test the current model on an unseen set. The losses generated by training and validation set over the 15 epochs can be seen in Figure 3. Since it appears that most of the learning is done in the first few epochs before the model starts overfitting the training data, early stopping was applied. The model with the lowest validation set loss was kept and evaluated on the yet unseen test set.

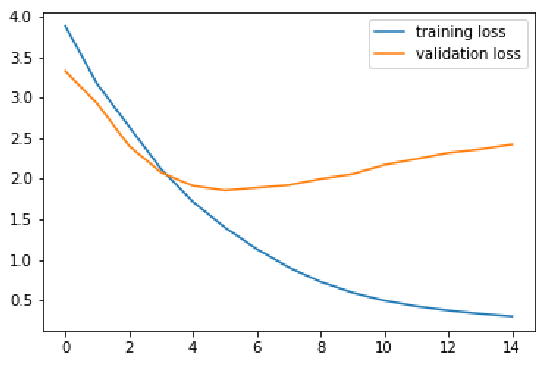


Figure 3: Training and validation loss of the initial baseline transformer model over 15 epochs. The model starts to overfit the training data after approximately 5 epochs.

This initial model served as the baseline to compare each other model to when moving along to other experiments. The initial model was able to achieve a final test loss of 2.5 and Bleu score of $b = 31.7$. This already represents a good model, as Bleu Scores above 30 are considered to reflect understandable translations (Lavie, 2011).

3.3 Hyperparameter Tuning

The best hyperparameters found by the 'Attention is All you Need" paper serve as a good baseline for further experimentation. But, because my transformer architecture is probably not exactly the same, as well as the fact that a different data set was used to train the transformers, it goes to reason that a different set of hyperparameters could achieve a better result. In order to test this hypothesis and improve the baseline transformer model, an experiment was done to tune the hyperparameters.

Because it is not always immediately clear how a different hyperparameter value can affect model performance, a grid search approach was chosen to find the best set of parameters. In this technique, many models are trained using a large variety of hyperparameter combinations. For every possible combination of hyperparameters, a new model was trained for 5 epochs. Once again, early stopping was applied and the model with the best validation loss during those 5 epochs was chosen as the best model of the run. This best model was then evaluated on a subsection of 200 randomly chosen test samples. These choices, only training for 5 epochs as well as evaluating the models on a subsection of the test data, were made in order to reduce the search time. Since these experiments were done using only a single GPU on my personal computer, minimizing resources spent was unfortunately a priority.

Overall, 48 models were trained and evaluated using the following hyperparameter combinations:

- d_{model} = [256, 512, 1024]
- n_{heads} = [8, 16]
- α = [0.003, 0.0001]
- n_{layer} = [1, 2, 3, 6]

Out of these, the combination with the best Bleu Score on the test set sample was saved and deemed to produce the best possible model. This combination was found to be d_{model} = 1024, n_{heads} = 8, α = 0.0001 and n_{layer}=6. The training and validation loss graphs for the 48 trained models during this experiment have been omitted from the report for the sake of clarity.

3.4 Manual Tuning

In order to continue improving the model, the best hyperparameters found in section 3.3 were then manually tuned. Interestingly, the grid search found that the best model uses the maximum value of each parameter, except for the number of multi-head-attention heads n_{heads}. Now, the question was whether increasing these other values further would actually lead to a better model. Unfortunately, it was not possible for me to increase the embedding size, as this would lead to my GPU running out of memory.

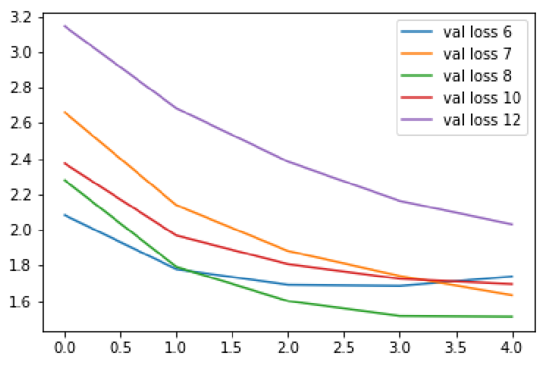


Figure 4: Validation loss of the manual tuning experiments using a varying number of encoder/decoder layers over 5 epochs.

I therefore decided to train a few more models with an increased number of layers. Here I once again trained the models for 5 epochs each, keeping the epoch with the best validation loss as the best model. I evaluated these best models per epoch on a random sample of 200 test set sentence pairs in order to calculate test loss and Bleu Score. The model with the best Bleu Score was then considered to have the best possible hyperparameters.

The validation loss results of these runs can be found in Figure 4. We can see that the model performance, measured on low validation loss, does improve initially when going from 6 to 8 layers. Going even further, though, the performance starts to worsen again. The same result can be seen in Table 1, which shows the Bleu Scores for these experiments. It was therefore decided that a model with n_{heads} = 8, together with the parameters found in section 3.3, represents the sweet spot and would produce the best machine translation model.

n_{heads}	Bleu Score
6	29.9
7	31.5
8	34.2
10	32.6
12	30.5

Table 1: Result of Bleu Score on a random sample of test data during manual hyperparameter tuning

3.5 Final Model Result

After finding the best hyperparameter combination using a grid search technique over many combinations of hyperparameters described in section 3.3 and further manual tuning describes in section 3.4, a final machine translation model was trained. This model used the best found hyperparameters and was trained over 15 epochs, just like the initial translation model ins section 3.2. These hyperparameters where d_{model} = 1024, n_{heads} = 8, α = 0.0001, d_{ff} = 4096 and n_{layer}=8. The result of the training, including training and validation loss can be seen in Figure 5.

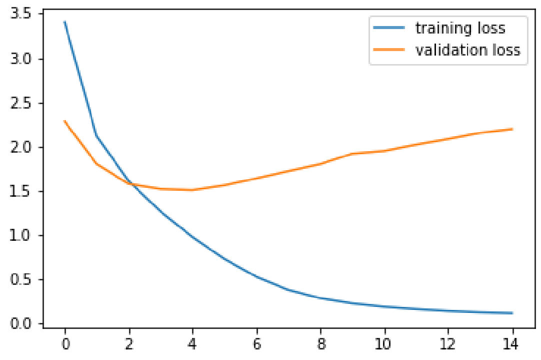


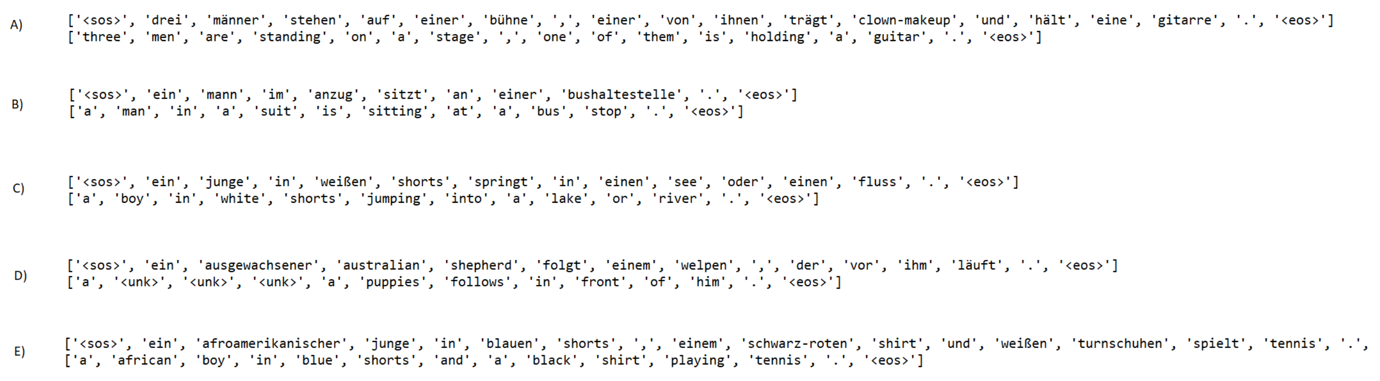
Figure 5: Training and validation loss of the final transformer model over 15 epochs. The model starts to overfit the training data after approximately 3 epochs.

Here, the training and validation loss was recorded after every epoch. It is clear that the model learns sentence translations over time, as the average training loss decreases with every epoch. The average validation loss, on the other hand, initially decreases, but then quickly starts to increase again. Once again, the model starts overfitting on the training data. After only approximately 3 epochs, the validation loss starts to increase while the training loss further decreases, meaning that overfitting is the most likely cause. In future work, it might be a good idea to further regularize the model, or to adjust the dropout rate, to try and reduce this effect. In order to find the best possible final mode, early stopping was applied at the model with the best validation loss. This final model was then evaluated on the unseen test set, achieving a final test loss of 1.51 and Bleu Score of $b = 39.4$.

3.6 Sample Sentences

After the final model had been trained, I used it to translate the entire 1000 sentence pairs contained in the test set. I wanted to take this opportunity to show off some of these translations, in order to give the reader a feeling for how well the final machine translation model would perform in practice. These samples can be seen in Figure 6.

Looking more closely at Figure 6, we can see that even complicated sentences are translated fairly well, such as A) or E). Some words are omitted, but the overall translation is understandable and conveys the meaning of the sentence. Even when some words are unknown to the mode, the resulting translation is still quite good, like in example D). Here, the words ”asugewachsener australian



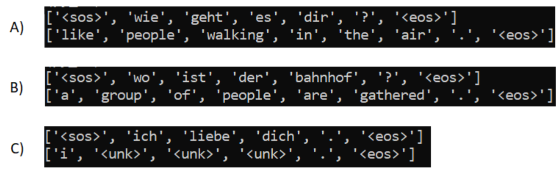
```
A)    ['<sos>', 'drei', 'männer', 'stehen', 'auf', 'einer', 'bühne', ',', 'einer', 'von', 'ihnen', 'trägt', 'clown-makeup', 'und', 'hält', 'eine', 'gitarre', '.', '<eos>']
      ['three', 'men', 'are', 'standing', 'on', 'a', 'stage', ',', 'one', 'of', 'them', 'is', 'holding', 'a', 'guitar', '.', '<eos>']

B)    ['<sos>', 'ein', 'mann', 'im', 'anzug', 'sitzt', 'an', 'einer', 'bushaltestelle', '.', '<eos>']
      ['a', 'man', 'in', 'a', 'suit', 'is', 'sitting', 'at', 'a', 'bus', 'stop', '.', '<eos>']

C)    ['<sos>', 'ein', 'junge', 'in', 'weißen', 'shorts', 'springt', 'in', 'einen', 'see', 'oder', 'einen', 'fluss', '.', '<eos>']
      ['a', 'boy', 'in', 'white', 'shorts', 'jumping', 'into', 'a', 'lake', 'or', 'river', '.', '<eos>']

D)    ['<sos>', 'ein', 'ausgewachsener', 'australian', 'shepherd', 'folgt', 'einem', 'welpen', ',', 'der', 'vor', 'ihm', 'läuft', '.', '<eos>']
      ['a', '<unk>', '<unk>', '<unk>', 'a', 'puppies', 'follows', 'in', 'front', 'of', 'him', '.', '<eos>']

E)    ['<sos>', 'ein', 'afroamerikanischer', 'junge', 'in', 'blauen', 'shorts', ',', 'einem', 'schwarz-roten', 'shirt', 'und', 'weißen', 'turnschuhen', 'spielt', 'tennis', '.',
      ['a', 'african', 'boy', 'in', 'blue', 'shorts', 'and', 'a', 'black', 'shirt', 'playing', 'tennis', '.', '<eos>']
```

Figure 6: Sample translations generated using the final translation model from German sentences taken out of the test set.

shepherd" are not a part of the model vocabulary, but they are correctly replaced with the "unknown" token.

3.7 Custom Sentences

In addition to the test set translations, I also created a program that allows a user to translate customs sentences using the final model. Using this program, I tested the final model on some custom German sentences. Interestingly, the translation model does quite well when it has to translate descriptions, e.g. "Der Himmel ist Blau" $\rightarrow$ "the sky is blue.". But, the model fails to translate sentences correctly when confronted with more natural language, i.e. questions.



```
A)  ['<sos>', 'wie', 'geht', 'es', 'dir', '?', '<eos>']
    ['like', 'people', 'walking', 'in', 'the', 'air', '.', '<eos>']

B)  ['<sos>', 'wo', 'ist', 'der', 'bahnhof', '?', '<eos>']
    ['a', 'group', 'of', 'people', 'are', 'gathered', '.', '<eos>']

C)  ['<sos>', 'ich', 'liebe', 'dich', '.', '<eos>']
    ['i', '<unk>', '<unk>', '<unk>', '.', '<eos>']
```

Figure 7: Translations of questions generated using the final translation model. Even simple questions like "how are you?" are not translated correctly (A).

Simple questions like "wie geht es dir?" ("how are you?") or "wo ist der bahnhof?" ("where is the train station?") are translated into nonsense. This is mostly likely due to the training data set being translations of image descriptions. Since no conversational sentences are present int he training data, the model has not learned how to translate them.

4 Discussion

The experiments presented in this report show that the transformer architecture created during the course of this software project is able to learn from training data and improve over time. The hyperparameter optimizations done in Section 3.3 and Section 3.4 further show that the model can be tuned to produce high quality translations. The final result produces translations that are readable and understandable, even when more complex scenarios are presented as shown in Section 3.6. I therefore believe that this software project was very successful, and a very good learning experience. Still, there are of course some improvements to the model that can still be made.

4.1 Possible Improvements

Although the final results are promising on the test data, there is one overarching problem that continues to show itself throughout the experiments: overfitting. The trained models all very quickly overfit the training data. One possible solution that could improve translation performance even further would be to introduce more regularization. It might be interesting to experiment with different regularization methods, or to simply increase the dropout percentage. By allowing the model to not overfit so quickly, the resulting translations and Bleu Score might improve even further. This problem is exacerbated by another weakness of the software project. That is the very small data set in comparison to the one used in the attention paper. We only have 29000 training pairs, while the attention paper uses a data set with 4.5 million sentences. Of course, this huge amount of data would not have been possible to process given the available tech-

nical resources. Still, using a larger data set might have produced a better result and helped to reduce overfitting.
Another aspect to consider is that the model only performs so well on sentences that are similar to the ones found in the Multi30k data set. That is, sentences that read like image descriptions. As shown in Section 3.7, the model fails to produce correct translations when confronted with questions. This intuitively makes a lot of sense, since the model obviously can only learn from the data it is given. But, it is an aspect that must be kept in mind: the good translations on image descriptions do not translate into other areas. Once again, a larger, more diverse data set would have helped to solve the problem.

5 Conclusion

Overall, the final translation model performs very well. The sample sentences shown in section 3.6 illustrate this point quite effectively. Complicated sentences from the data set are turned into understandable translations. Even when unknown words are present, the meaning and structure of the sentence are preserved. This is also reflected in the final Bleu Score of 39.2. A Bleu Score of almost 40 can already be considered as belonging to a model that produces high quality translations. Such a score even beats most of the translation models presented in the ”Proceedings of the First Conference on Machine Translation”, where the same data set was used (Specia et al., 2016).

Comparing it to the original ”Attention is All You Need” paper yields similar results. Of course, here the translations were done on a different data set and using different language pairs, but the point still stands. The model produced during this software project yields good translations, and I was able to train it at home on my own. I think that this is a great testament to the power of the attention mechanism.

References

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Alon Lavie. 2011. Evaluating the output of machine translation systems.

Lucia Specia, Stella Frank, Khalil Sima’an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 543–553, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.