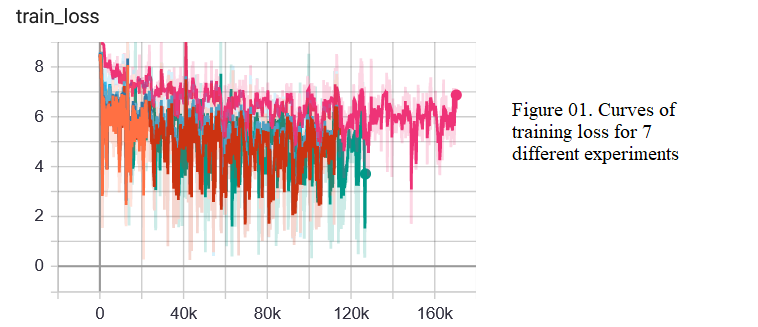
# Fine-tune a pretrained T5 transformer & Building a transformer from scratch

T5 transformer, standing for “text-to-text transfer learning,” was originally proposed in the paper [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](https://arxiv.org/pdf/1910.10683.pdf). The goal of T5 is to (1) explore the landscape of transfer learning for natural language processing. (2) provide a unified framework that converts all text-based problem to a text-to-text format. Our goal was to investigate how well T5 can be fine-tuned for English-Tibetan translation, a task for a low-resource language.

We have provided only one piece of code “T5.py” as an example of our training pipeline. However, it took several steps and trials-and-errors before reaching this final product.

## Stage 1: Initial experiment

Initially, we used Pytorch-Lightning (a wrapper on top of Pytorch) and imported pretrained T5 model from huggingface transformers. We experimented with different learning rates (3e-4, 1e-4, 1e-5), learning rate schedulers (linear decay and constant), and different numbers of epochs (2, 10, 15). We acquired the first impression about the quality of model by observing the shape of learning curve. Unfortunately, adjusting the learning rates and the number of epochs did not change the fact that the learning curve converged at a high value. Figure 01 shows the learning curves of seven different experiments covering the tuning of hyperparameters above. The curves have similar patterns of oscillations. Further, the model ended up predicting very similar translations for different texts.



## Stage 2: Reading papers and blogs

As adjusting learning rates and number of epochs are the most straight-forward approaches and they did not seem to work, we realized that there were mechanisms of transformers or transfer learning that we were not utilizing correctly. We thoroughly read the T5 paper and several posts about transformers. Most of the information we learned was either too technical to understand or too broad to provide insight for model tuning.

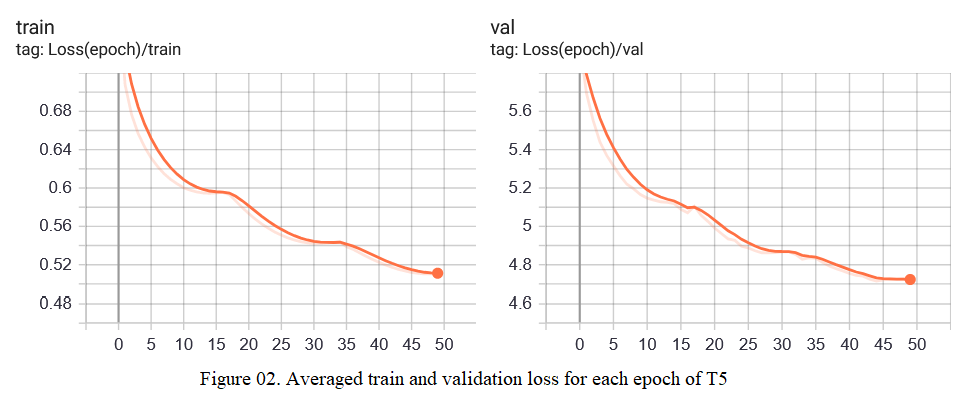
Thankfully, we have learned some key details that improved our training by certain extent—appropriate addition of special tokens <s></s><pad>. The target sentences fed to a transformer model need to be shifted to the right by a “begin-of-sentence” (BOS) token denoted as <s>, and then finish by an “end-of-sentence” token </s>. Further, a better use of <pad> token is to pad the token vectors in the same training batch to a given length, instead of padding all vectors to the same length (like we did in stage 1). To retrieve batch of token ids with different lengths, we needed to define a data iterator by ourselves that was not supported by Pytorch-Lightning. Thus, we decided to switch to Pytorch native.

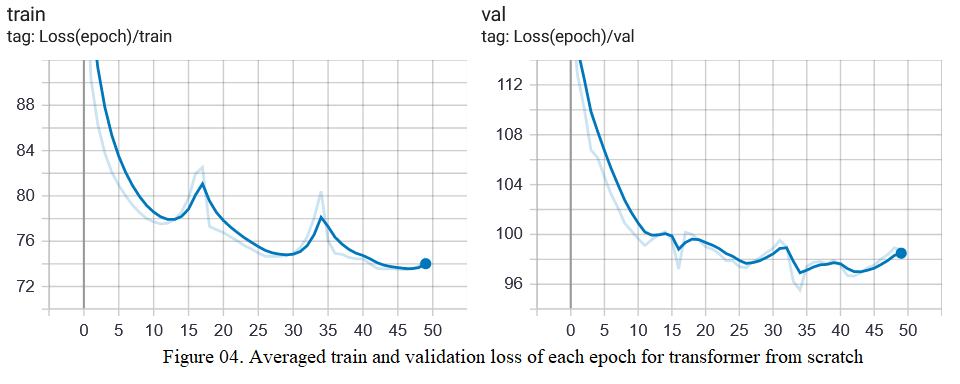
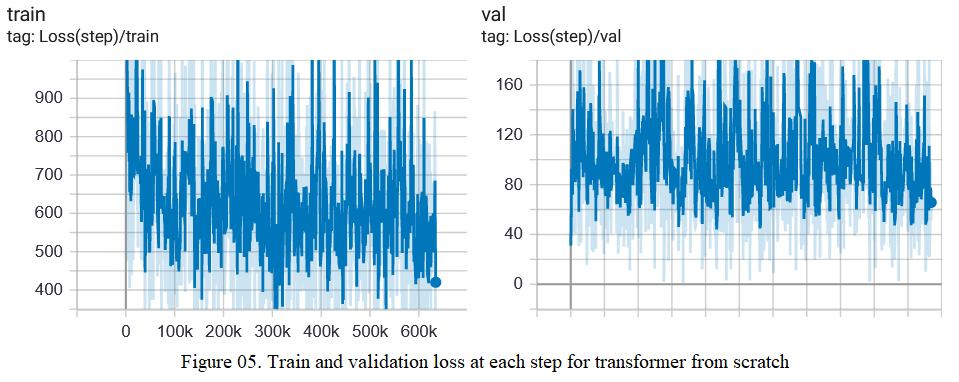
## Stage 3: Second round of experiment

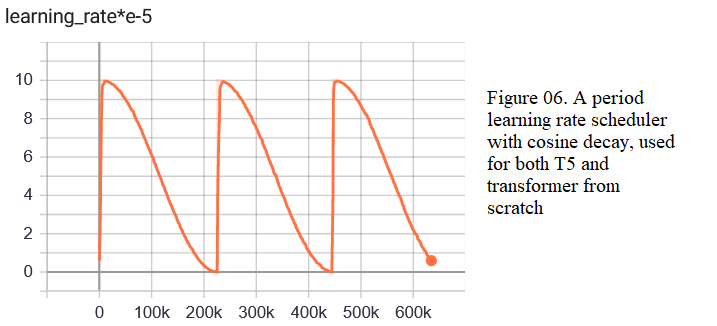
In this round of experiments, we incorporated the following changes:

* Pad each token vector to the maximum length in the current batch instead to a uniform length.
* Train up to 50 epochs and save the model when the validation loss is the lowest. The purposes were (1) prioritize solving underfitting before overfitting; (2) observe a more holistic trend of training curve; (3) get an idea when the model begins to significantly overfit.
* Aside from T5 model, we also built a transformer from scratch.
* Add special tokens <s></s><pad> to target sentences appropriately.
* Used a scheduler periodically spikes to 1e-4 and decay according to a cosine function.
* Introduced dropout and weight decay.

The following figures show the recorded information during training.







Here are several observations with the new experiments:

* Fine-tuning a T5 model result in lower losses than training a transformer from scratch. This makes sense because the purpose of pretraining is to adapt a model for sequence-to-sequence tasks. As optimization for deep learning is usually non-convex, a model that is not pretrained tends to be stuck at local minima, resulting in high loss.
* The translation generated by new model is still far from being good. However, by comparing sample translations (e.g. “T5\_sample\_results.txt” and “Scratch\_sample\_results.txt”), we got the general impression that the quality of translation by T5 is slightly higher than the transformer from scratch. In addition, after correctly processing the
* It is surprising how the transformer architecture from FairSeq library performed better than both our models written in Pytorch and hugging transformers.