The Challenges of using Transformers in ASR

Posted on January 8, 2020, 11 minute read

Since mid 2018 and throughout 2019, one of the most important directions of research in speech recognition has been the use of self-attention networks and transformers, as evident from the numerous papers exploring the subject. In this post, I try to provide an overview of the major challenges when using transformers for ASR, and the different solutions proposed in the several papers.

Preliminary: I assume that the reader is familiar with self-attention and transformers, first proposed in this paper (https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf). For a quick refresher, I highly recommend The Illustrated Transformer (http://jalammar.github.io/illustrated-transformer/).

Here are the problems I will discuss in this post:

1. Long sequence length
2. Positional encoding
3. Difficult to train
4. Large model size
5. Streaming ASR
6. Parallelization

---

1. Long sequence length

Transformers were originally proposed for machine translation, where sequence lengths are short (~ 40 words on average). Sequence lengths in ASR, on the other hand, are of the order of a few thousand frames. Since self-attention encoder blocks have quadratic computational complexity, this means that computing the self attention between all pairs of frames is too expensive. Moreover, individual frames in a speech sequence are less informationally dense than words in a sentence (in case of MT). Therefore, almost all work exploring transformers for ASR use some tricks to deal with the sequence length problem.

- Povey et al.[1] first proposed **time-restricted self-attention** which computes self-attention between pairs of frames within a fixed sized context window, rather than all the frames in the utterance. Yeh et al.[2] also employed a similar strategy and called it **truncated self-attention**.

- Several papers use some kind of *downsampling* approach to reduce the sequence length prior to passing it to the self-attention layer. Sperber et al.[3] first proposed downsampling based on **reshaping** operation in the context of an LAS model, and this was replicated in Pham et al.[4].

- Use of **convolutional layers** is very common for downsampling, and this has been effectively employed in Dong et al.[5], Bie et al.[6], and Tsunoo et al.[7].

- Salazar et al.[8] also compared downsampling with **subsampling** and **pooling** operations, in addition to reshaping, for a CTC-based end-to-end model. Dong et al.[9] also used temporal pooling for downsampling.

---

2. Positional encoding

Since self-attention networks are inherently position agnostic, we need to explicitly encode frame positions in the model. The original paper used sinusoidal position encodings for this purpose, but several alternatives have been explored in the ASR context.

- Dong et al.[5], Zhou et al.[10], Salazar et al.[8], and Tsunoo et al.[7] used **sinusoidal positional encodings** directly as was proposed in the original Transformer paper. Bie et al.[6] also analyzed sinusoidal positional encodings and found that they hurt performance because of longer sequences at test time. Sperber et al.[3] argued that this is because "inputs are fixed feature vectors rather than trainable word embeddings, making it difficult for the model to separate position and content for each state".

- To avoid the problem of tail deletion due to sinusoidal encodings, Zhou et al.[11] used **relative positional embeddings** which is added to the similarity computation in the self-attention. Sperber et al.[3] had also previously experimented with variants of this method. They also tried using **RNN layers** stacked or interleaved in the encoder to provide positional information.

- Perhaps the simplest way to add position information is to append **one-hot relative positional encodings** with the input. This was used in Povey et al.[1].

- The most popular approach recently is to use **convolutional layers**, since this provides the twin benefits of downsampling as well as positional encoding. This technique has been used in Bie et al.[6], Yeh et al.[2], and Mohamed et al.[12]. Wang et al.[13] compared the use of sinusoidal encodings, frame stacking, and convolutional layers for transformers in a hybrid model, and found that convolutional layers work best.

---

3. Difficult to train

It has been found that in order to achieve good performance with Transformers, both in hybrid and end-to-end systems, we need the "self-attention+FFN" architecture and a deep network. However, this also makes the training more difficult since the model is more likely to get stuck at a local optimum. Several methods have been proposed to prevent this problem.

- Wang et al.[13] proposed **iterated loss**, which uses the outputs at several intermediate layers in addition to the last layer to calculate the auxiliary cross-entropy loss. Interpolation of all such losses gives the final objective function to be minimized.

- Pham et al.[4] proposed a variation of **stochastic depth** inspired by Stochastic Residual Networks (https://arxiv.org/abs/1603.09382). This is similar to dropout, where some layers are randomly dropped during training.The authors found that a policy where the drop probability is proportional to the layer depth works best in practice.

---

4. Large model size

Another problem that comes with the use of very deep Transformers is the large model size, especially when we need to build on-device ASR systems. This has been addressed in different contexts.

- Bie et al.[6] showed that similar performance can be obtained using a simplified model, while also compressing the model 4x by fully **quantizing to 8-bit** fixed point precision.

- **Weight matrix factorization** is a general technique which is used for reducing model complexity. In this approach, a large weight matrix W is expressed as the product of two smaller matrices A and B. Often, one of these matrices is constrained to be orthonormal, as in the TDNN-F layers used in Kaldi. Winata et al.[14] proposed the low-rank Transformer by using similar principles and found that up to 50% parameter reduction can be obtained with better validation and test error rates using this method.

---

5. Streaming ASR

In the conventional Transformer architecture, self-attention is computed between every pair of frames in the sequence. This means that the encoder must see the entire utterance before processing can start. Often, we need the process to be "streaming", meaning that transcription must be obtained as soon as something is spoken, or with very low latency. To address this, several modifications have been proposed to the model.

- Wang et al.[13] experimented with using a **limited right context** during inference, when using the Transformer for acoustic modeling in a hybrid system. They found that although this creates a large mismatch between training and inference, the resultant systems can still yield reasonable WERs if the number of right context frames is large enough. A similar approach (causal convolutions and truncated self-attention) was also employed by Yeh et al.[2], but used both in training and inference.

- Tsunoo et al.[7] proposed **contextual block processing** to make the encoder streamable. In this approach, the utterance is divided into chunks and processed in parallel. A context vector learned from each chunk is fed as additional input to the following chunk, for every layer in the encoder. In a follow-up paper[15], they proposed a method inspired by **monotonic chunkwise attention** to make the decoder streamable, thus making the entire model fit for online ASR.

---

6. Parallelization

- In sequence transduction problems such as ASR, there is a mismatch between training and inference phases when using autoregressive models. During training, the output is conditioned on the ground truth (teacher forcing), but at inference, it is conditioned on the prediction. To remedy this, a method called scheduled sampling is used which increases the probability of conditioning on the decoded hypotheses as training advances. However, this means that training cannot be done in parallel since we need the previous output. Zhou et al.[11] proposed two approaches for **parallel scheduled sampling** - with hybrid model result, and with self-decoding result. The idea is to acquire the whole input token series of decoder in advance by simulating the error distribution of inference.

- Chen et al.[16] proposed the **non-autoregressive transformer** by replacing the previous history with masked output. They also experimented with different decoding strategies, namely easy-first (update `mask` tokens with those with high confidence) and mask-predict (replace low confidence outputs with `mask` tokens).

- Salazar et al.[8] used **CTC with self-attention** instead of an encoder-decoder architecture. This made the model non-autoregressive and thus parallelizable at inference.

Here are some key insights that I gleaned from reviewing all this literature:

1. Using a few (typically 2) convolutional layers at the start seems to be useful both as a means of encoding positional information as well as to downsample the sequence. If streaming ASR is required, causal convolutions may be used.

2. To achieve good performance, very deep models are needed. However, they must be trained with some modified loss functions like the iterated loss.

3. For encoder-decoder models, either a parallel version of scheduled sampling is required to make training match inference, or the model must be made non-autoregressive with some kind of masking approach. It might be interesting to see if parallel scheduled sampling can be used in conjunction with the non-autoregressive transformer.

References:

1. D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, “A Time-Restricted Self-Attention Layer for ASR (https://www.danielpovey.com/files/2018_icassp_attention.pdf),” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, 2018. ↩ ↩2

2. C.-F. Yeh et al., “Transformer-Transducer: End-to-End Speech Recognition with Self-Attention (https://arxiv.org/pdf/1910.12977.pdf),” ArXiv, Oct. 2019. ↩ ↩2 ↩3

3. M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, “Self-Attentional Acoustic Models (https://arxiv.org/pdf/1803.09519.pdf),” ArXiv, Jun. 2018. ↩ ↩2 ↩3

4. N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, “Very Deep Self-Attention Networks for End-to-End Speech Recognition (https://arxiv.org/pdf/1904.13377),” ArXiv, May 2019. ↩ ↩2

5. L. Dong, S. Xu, and B. Xu, “Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition (http://150.162.46.34:8080/icassp2018/ICASSP18_USB/pdfs/0005884.pdf),” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, 2018. ↩ ↩2

6. A. Bie, B. Venkitesh, J. Monteiro, M. A. Haidar, and M. Rezagholizadeh, “Fully Quantizing a Simplified Transformer for End-to-end Speech Recognition (https://arxiv.org/pdf/1911.03604),” ArXiv, Nov. 2019. ↩ ↩2 ↩3 ↩4

7. E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, “Transformer ASR with Contextual Block Processing (https://arxiv.org/pdf/1910.07204),” ArXiv, Oct. 2019. ↩ ↩2 ↩3

8. J. Salazar, K. Kirchhoff, and Z. Huang, “Self-Attention Networks for Connectionist Temporal Classification in Speech Recognition (https://arxiv.org/pdf/1901.10055),” ICASSP 2019 - 2019 IEEE International

Conference on Acoustics, Speech and Signal Processing (ICASSP). ↩ ↩2 ↩3

9. L. Dong, F. Wang, and B. Xu, “Self-Attention Aligner: A Latency-Control End-to-End Model for ASR Using Self-Attention Network and Chunk-Hopping (https://arxiv.org/pdf/1902.06450.pdf),” ArXiv, Feb. 2019. ↩

10. S. Zhou, L. Dong, S. Xu, and B. Xu, “Syllable-Based Sequence-to-Sequence Speech Recognition with the Transformer in Mandarin Chinese (https://arxiv.org/pdf/1804.10752),” ArXiv, Jun. 2018. ↩

11. P. Zhou, R. Fan, W. Chen, and J. Jia, “Improving Generalization of Transformer for Speech Recognition with Parallel Schedule Sampling and Relative Positional Embedding (https://arxiv.org/pdf/1911.00203),” ArXiv, Nov. 2019. ↩ ↩2

12. A. Mohamed, D. Okhonko, and L. Zettlemoyer, “Transformers with convolutional context for ASR (https://arxiv.org/pdf/1904.11660),” ArXiv, Apr. 2019. ↩

13. Y. Wang et al., “Transformer-based Acoustic Modeling for Hybrid Speech Recognition (https://arxiv.org/pdf/1910.09799),” ArXiv, Oct. 2019. ↩ ↩2 ↩3

14. G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, and P. Fung, “Lightweight and Efficient End-to-End Speech Recognition Using Low-Rank Transformer (https://arxiv.org/pdf/1910.13923.pdf),” ArXiv, Oct. 2019. ↩

15. E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, “Towards Online End-to-end Transformer Automatic Speech Recognition (https://arxiv.org/pdf/1910.11871),” ArXiv, Oct. 2019. ↩

16. N. Chen, S. Watanabe, J. Villalba, and N. Dehak, “Non-Autoregressive Transformer Automatic Speech Recognition (https://arxiv.org/pdf/1911.04908.pdf),” ArXiv, Nov. 2019. ↩

Tags: speech recognition, transformer

← PREVIOUS POST (/2019-12-19-ASRU-PAPERS/)

NEXT POST → (/2020-05-18-USING-LIBRISPEECH/)

Stefan Falk • a year ago

Hi!

Thank you for this interesting article!

I have a question regarding positional encodings. Do you know why we never see something like this

```
# Embed our features
embedded = embedding(features)

# Run through an RNN i.o. to extract positional information
pos = rnn_stack(embedded)

# Use the embeddings and positional encodings in the transformer
x = transformer_encoder([embedded, pos])
```

I would kind of assume that the RNN-stack should learn the the missing information required by the transformer-encoder (intuitively speaking) but I usually see the other methods you described in your article. Is my intuition wrong here?

▲ | ▼ • Reply • Share ›

Desh Raj Mod ➚ Stefan Falk • a year ago

Hi Stefan, I'm glad you found the post useful. What you are suggesting is a totally valid architecture IMO, but it defeats the purpose of using self-attention, since the original idea wa do away with sequential processing required by RNNs. Also, it is a bit of an overkill, since we expect the self attention layers to learn most of the inter-dependence just using the positiona encoding (it is called an "encoding" and not an "embedding" for the very reason that it is not exactly "learnt" during the training). That said, it might be interesting to try out this architect see if it can outperform the usual approaches (it seems to have been used for neural MT in th paper: https://www.aclweb.org/anth....

▲ | ▼ • Reply • Share ›

Stefan Falk ➚ Desh Raj • a year ago

Hi! Thanks for your reply. What you say makes sense. However, I noted that positiona encodings can become a bit tricky when it comes to streaming. You also mentioned techniques like MoChA but, just implementation-wise speaking here, just using an RN seems so much simpler to me than other techniques.

If we look at TransformerXL it appears that they went back to "learning" these encodi well (see https://arxiv.org/pdf/1901.... Section 3.3). But I can't say that I fully underst how to implement this exactly.

But you are right with what you say in general, RNNs are not exactly very efficient in r ways.

Btw. the link to the paper you posted does not work for me :/

▲ | ▼ • Reply • Share ›

Desh Raj Mod ➚ Stefan Falk • a year ago

Sorry, can you check if this one works: https://www.aclweb.org/anth...

Positional encodings which work for text sequences may not always transfer w speech sequences IMO simply because of the difference in sequence lengths. Y streaming complicates things, and I don't fully understand the MoChA implementation myself, so RNN could indeed be a much simpler thing to try implementation-wise.

▲ | ▼ • Reply • Share ›

Stefan Falk ➚ Desh Raj • a year ago

Tell me about it :D

Thank you for the link. I am going to take a look at it. And thanks again for the article :)

▲ | ▼ • Reply • Share ›

(https://github.com/desh2608) (https://twitter.com/rdesh26) (mailto:draj@cs.jhu.edu)
(https://linkedin.com/in/rdesh26)